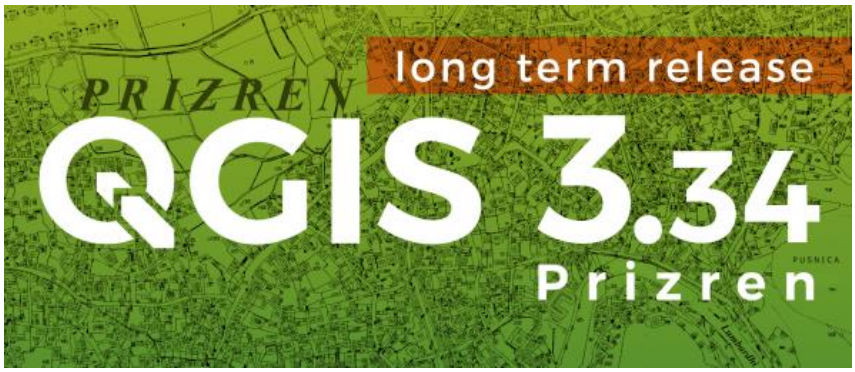
JEAN-MARC GILLIOT
QGIS-tuto.fr

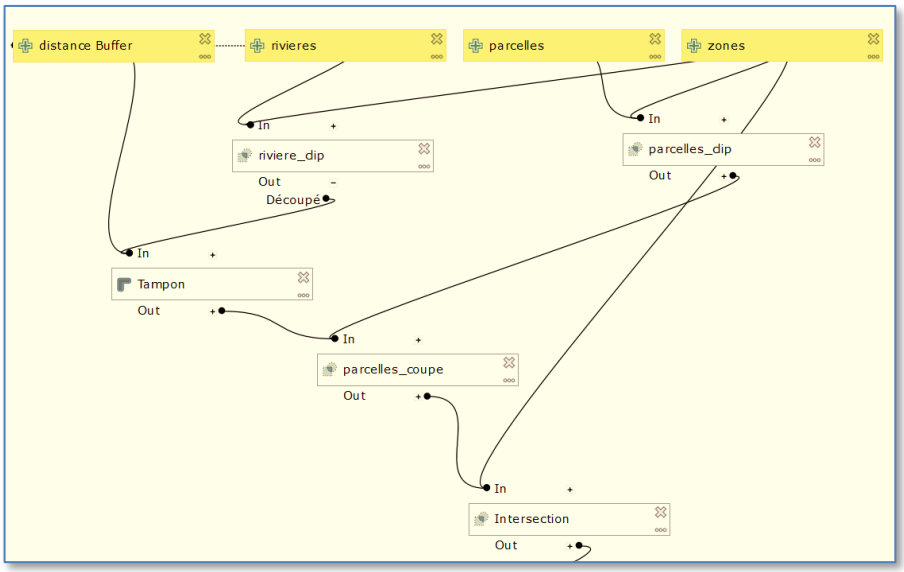
2024-2025

3 h 00 min

SIG
TD 6
QGIS 3.34

INTRODUCTION A L'AUTOMATISATION DANS QGIS : MODE BATCH, MODELEUR GRAPHIQUE, SCRIPT PYTHON et R





Version d'octobre 2024

AgroParisTech | université PARIS-SACLAY

Grande école européenne d'ingénieurs et de managers
dans le domaine du vivant et de l'environnement

Support de TD
QGIS 3.34

Jean-marc.gilliot@agroparistech

 Conseil  A vous de jouer  Attention  durée  objectif

clic souris : gauche  droit   tuto. Vidéo  à retenir

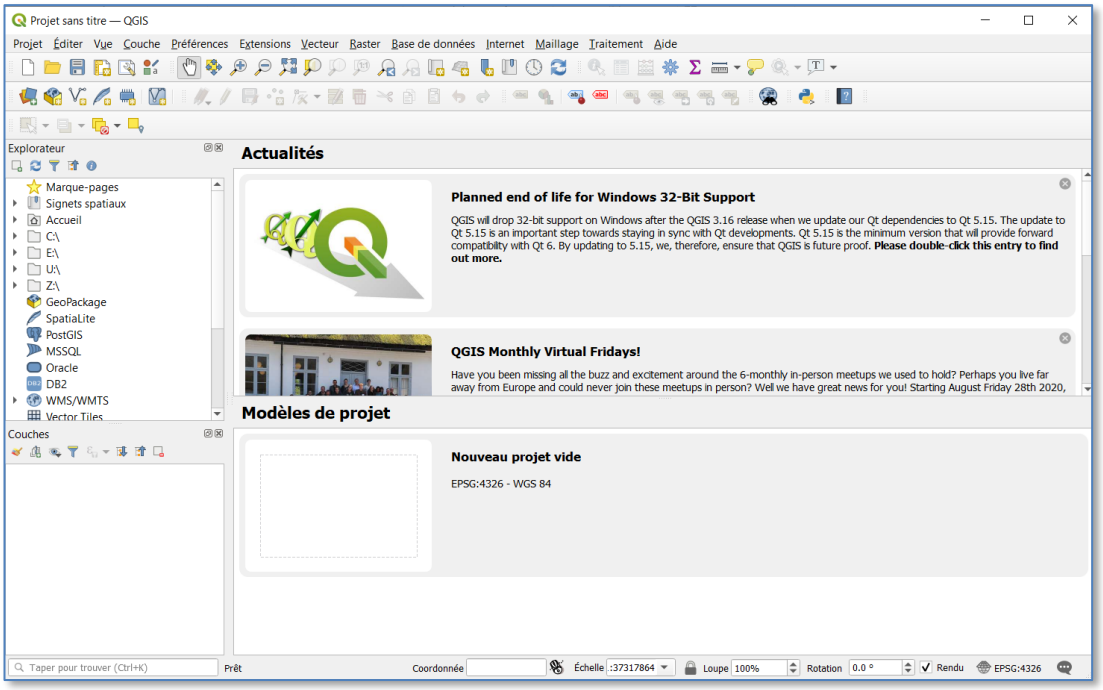
1. Démarrage de QGIS

Depuis le menu Windows  : le menu de QGIS est QGIS 3.34

Lancer ***QGIS Desktop 3.34*** 

https://qgis.org/downloads/

Sur le site de téléchargement de QGIS, privilégier la dernière version dite « LTR » (long time release) qui est la version la plus stable car elle subit des tests plus rigoureux avant sa sortie et reçoit au moins un an de mises à jour de correction de bogues.



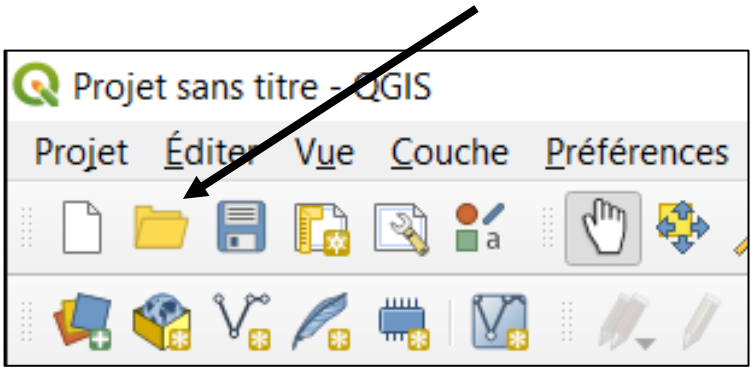
Télécharger le jeu de données « La Flèche » = fichier La_Fleche.zip

Décompresser le fichier ZIP dans votre disque de travail

Le dossier LA_FLECHE contient les données du TD

Depuis QGIS ouvrir le fichier de projet LA_FLECHE.qgs
qui est à la racine du dossier « LA_FLECHE »

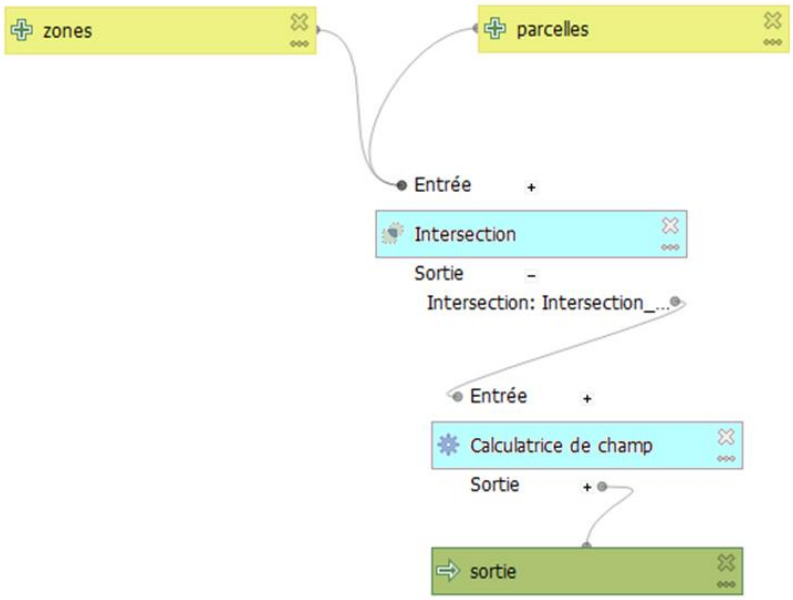
Menu → Projet → Ouvrir Ou le bouton ouvrir



On peut aussi simplement double cliquer sur la_fleche.qgs dans l'explorateur de fichiers Windows, pour lancer QGIS avec le projet, ou depuis le panneau explorateur de fichier QGIS

2. Chaine de traitements et traçabilité

Le plus souvent pour réaliser une analyse géographique sous SIG, on va enchainer une série d'opérations avec leurs entrées et leurs paramètres, on parle de « chaine de traitements » comme dans la figure ci-dessous :



On peut être amené à reproduire une même chaine de traitements, par exemple avec des données différentes, il est alors très difficile de garantir qu'on aura appliqué les mêmes paramètres que la première fois, pour pouvoir comparer les résultats obtenus avec des données différentes.

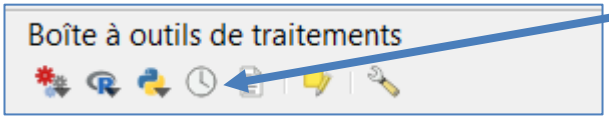
Il y a donc nécessité de garder la trace détaillée de toutes les opérations et de leurs paramètres afin de pouvoir les reproduire à l'identique : on parle de traçabilité. On peut noter dans un fichier texte toutes ces informations mais cela est très fastidieux.

Implémenter une chaine de traitement sous la forme d'un code informatique est une réponse efficace au besoin de traçabilité.

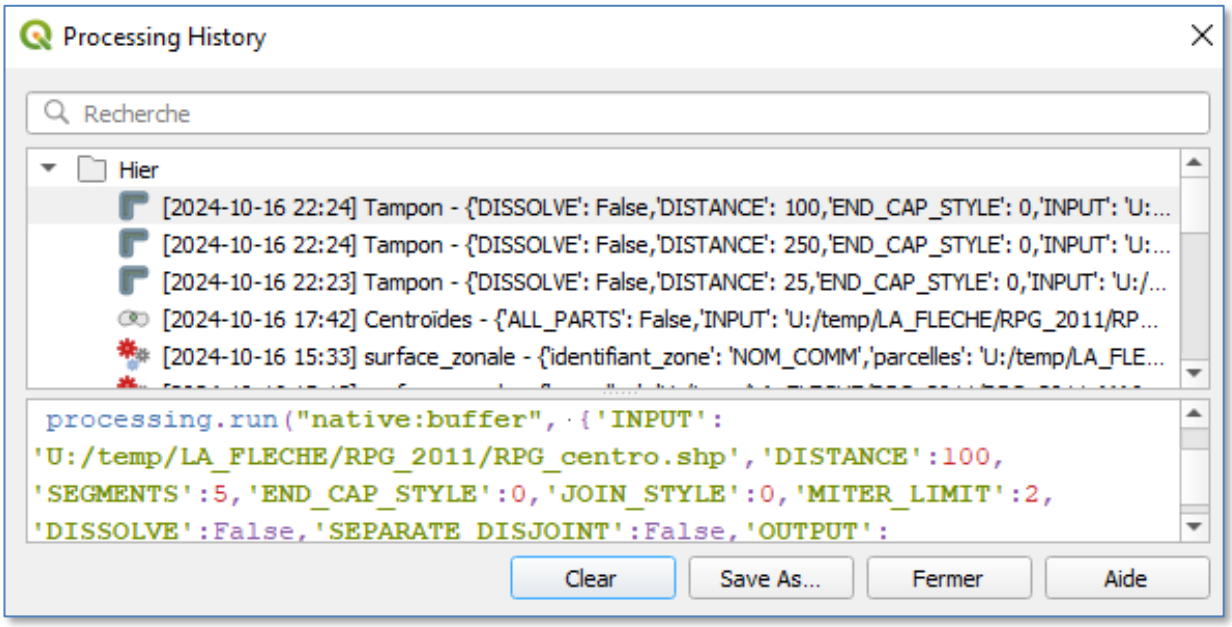
En effet on peut alors facilement réexécuter à l'identique un même code et donc reproduire à l'identique une chaine de traitement. Le code servira aussi de documentation pour conserver la mémoire de ce qui a été fait.

3. Historique de la boite à outils de traitement de QGIS

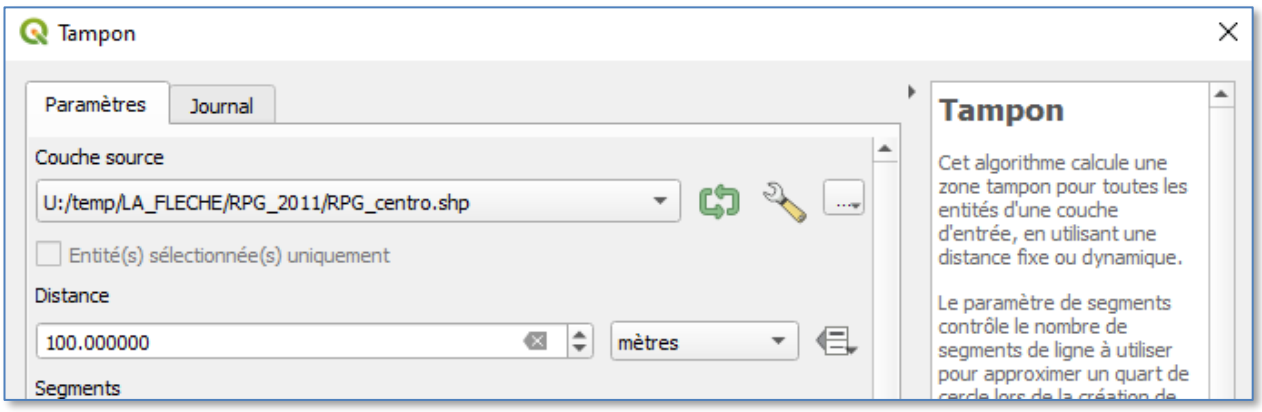
Un point très utile à connaitre sur la boite à outils de traitement de QGIS, c'est qu'elle gère un historique des traitements, accessible en cliquant sur 



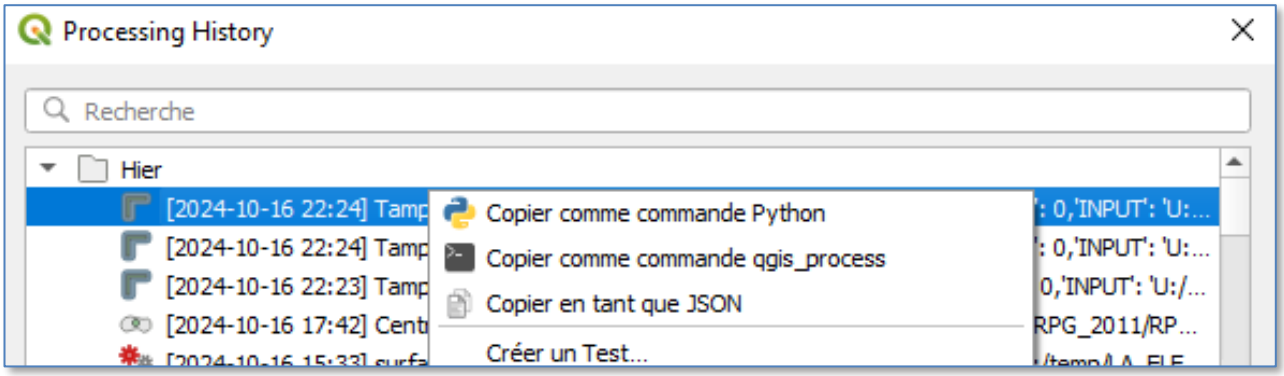
C'est la liste de toutes les dernières opérations effectuées dans la boite à outils, avec leurs paramètres, en cliquant sur une ligne on voit le détail de l'opération avec ses paramètres, plusieurs mois de traitements sont archivés.



En double cliquant sur la ligne, on relance la boite de dialogue de l'opération avec les mêmes paramètres.



Un clic droit sur une ligne permet de faire un copier/coller du code correspondent, dans différents langages :



Exemple de copier / coller en python :

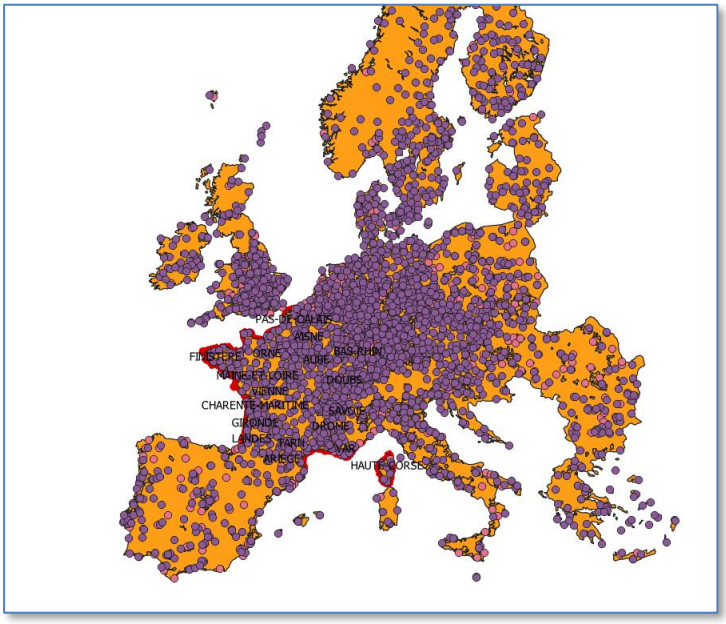
```
28
29  processing.run("native:buffer", {'INPUT':'U:/temp/LA_FLECHE/RPG_2011/
    RPG_centro.shp','DISTANCE':100,'SEGMENTS':5,'END_CAP_STYLE':0,'JOIN_STYLE':
    0,'MITER_LIMIT':
    2,'DISSOLVE':False,'SEPARATE_DISJOINT':False,'OUTPUT':'TEMPORARY_OUTPUT'})
30
```

4. Mode Batch (processus de lot) de QGIS

Le mode d'exécution « Batch » permet d'exécuter un même outil de traitement de manière automatique sur une série de fichiers d'entrée.

Affichez les couches :

LA_FLECHE → DIVERS → BATCH → airport_europ.shp
LA_FLECHE → DIVERS → BATCH → villes_europ.shp
LA_FLECHE → DIVERS → BATCH → dgur_europ.shp
LA_FLECHE → GEOFLA_FRANCE → FRANCE_FR



airport_europ
villes_europ
dgur_europ
FRANCE_FR

dgur = niveaux d'urbanisation

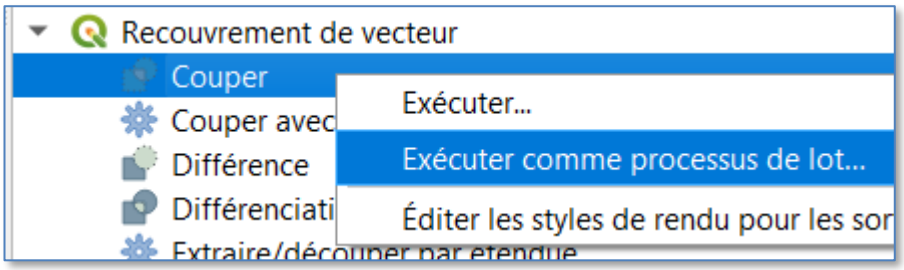
4.1 Le menu « processus de lot » dans la boite à outils de traitements

Découpez en une seule opération les 3 couches «xxx _europ » par le contour de la France en utilisant le mode « batch »

Utilisez la fonction « Couper » des outils de géotraitement

Recouvrement de vecteur → Couper

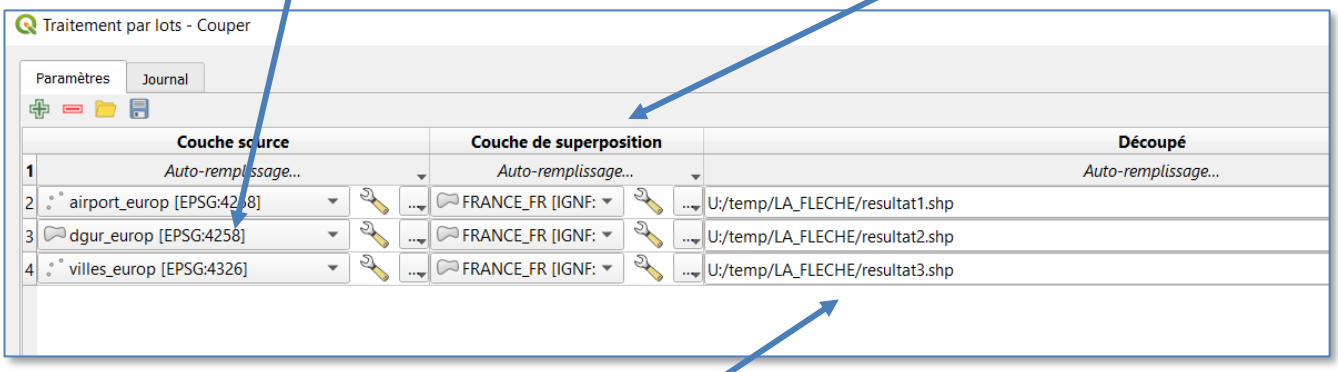
Par contre au lieu de la lancer comme d'habitude par un double clique bouton gauche souris, faire sur l'outil « Couper », puis choisir « Exécuter comme processus de lot »



Vous allez ajouter dans le tableau qui apparait une ligne pour chaque couche à couper par l'outil.

4.2. Remplissage du tableau des traitements et nommage automatique

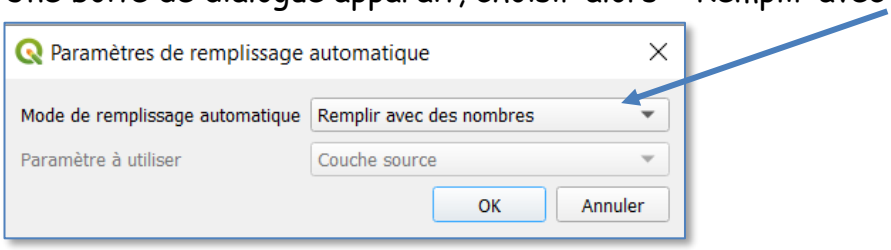
Créer 3 lignes avec le bouton et choisir comme couches sources : airport_eu, dgur_europ et villes_europ et comme couche de superposition France_FR



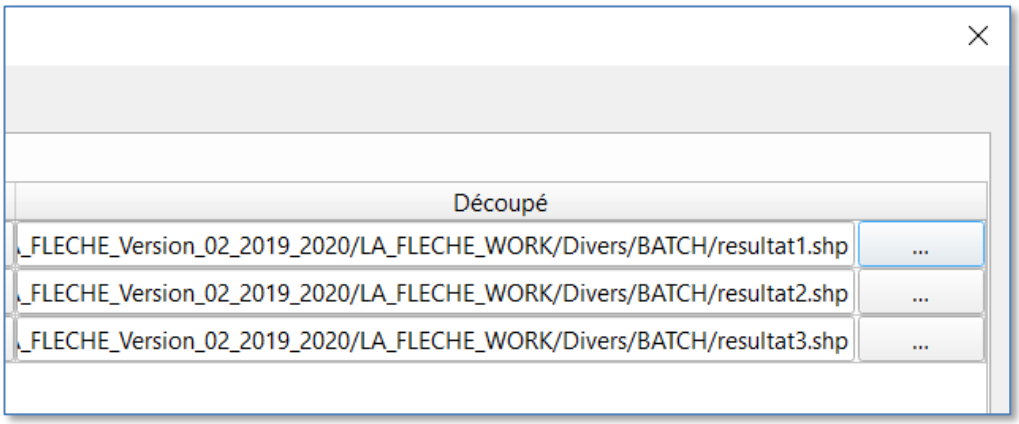
Découpé : ce sont les fichiers résultats

Cliquez sur ... sur la première ligne de Découpé puis donner un chemin vers un fichier de nom « resultat.shp » comme sortie dans un dossier (ici u:/temp)

Une boite de dialogue apparait, choisir alors « Remplir avec des nombres »

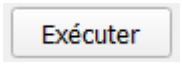


Des noms « automatiques » ont été générés, avec un nombre différent en fin.

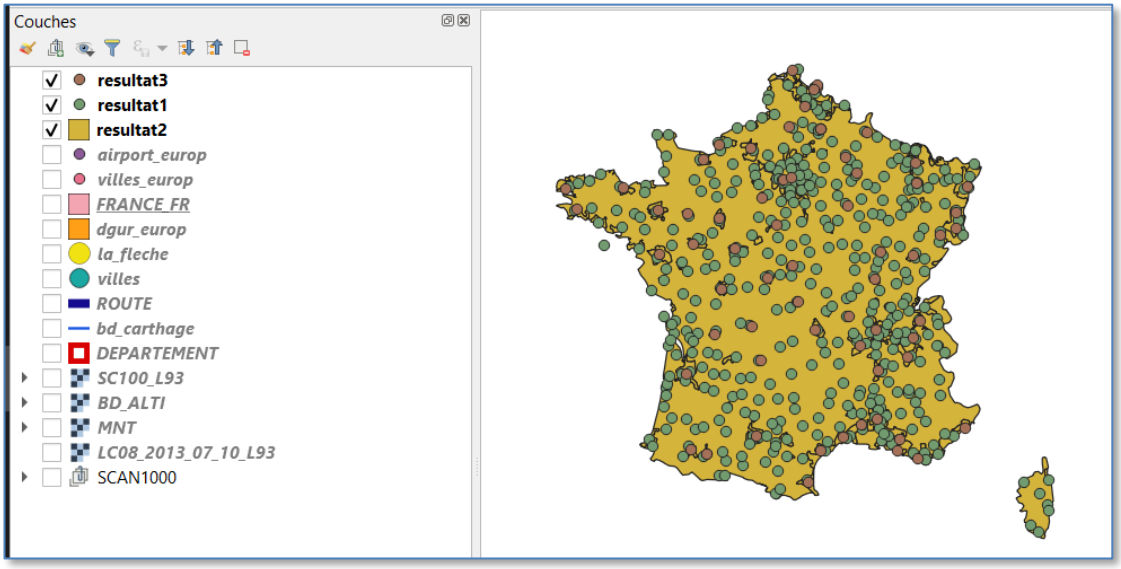


Sinon l'alternative est de choisir un nom manuellement pour chaque sortie.

Cochez  en bas à gauche de la boite de dialogue

Cliquez finalement sur 

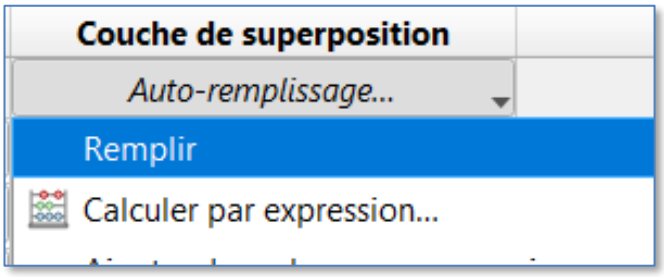
On a bien les 3 couches résultats qui sont découpées par le contour de la France.



Pour accélérer les traitements géographiques sur un grand nombre d'objet il est conseillé de préalablement calculer des index spatiaux avec :

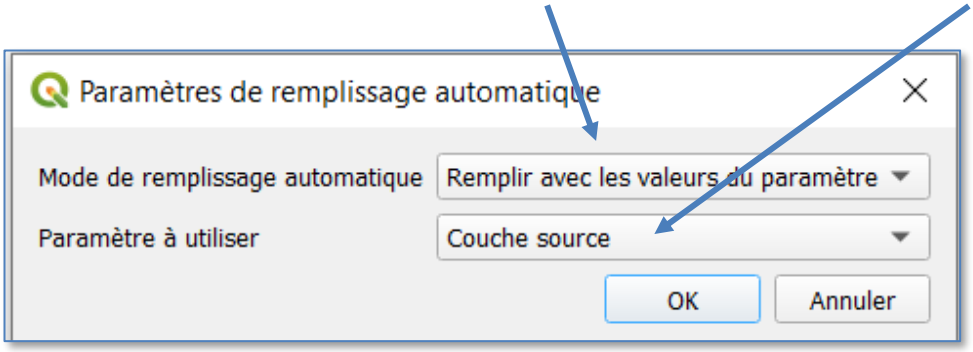
→ Outils généraux pour les vecteurs → Créer un index spatial

Si plusieurs lignes doivent avoir la même valeur, comme FRANCE_FR pour « couche de superposition », saisir la valeur sur la première ligne, puis cliquer sur « Auto remplissage » en haut de colonne et choisir dans le menu « Remplir » qui va recopier la valeur dans toutes les lignes suivantes (utiles quand on a un grand nombre de lignes).

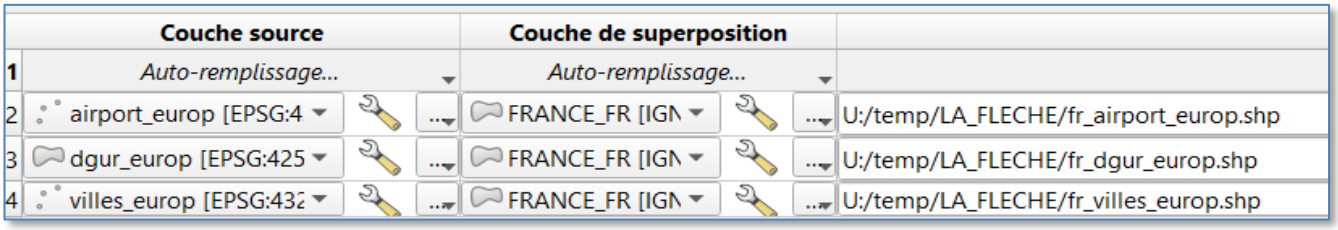


Refaire l'opération précédente mais en gardant pour les fichiers résultats un nom automatique qui soit lié au nom de la couche de départ

Répéter comme au point précédent en donnant comme nom de fichier résultat « fr_.shp », mais cette fois ci choisir comme mode de remplissage automatique :
« ***Remplir avec les valeurs du paramètre*** » et paramètre « ***Couche source*** »



Les couches résultats ont alors le même nom que la couche de départ préfixé par «fr_ »



Reprojetez en mode Batch, les 3 couches résultats en Lambert 93 (EPSG 2154), en créant des fichiers préfixés par « L93_ »

Outils généraux pour les vecteurs → Reprojeter une couche

À retenir :

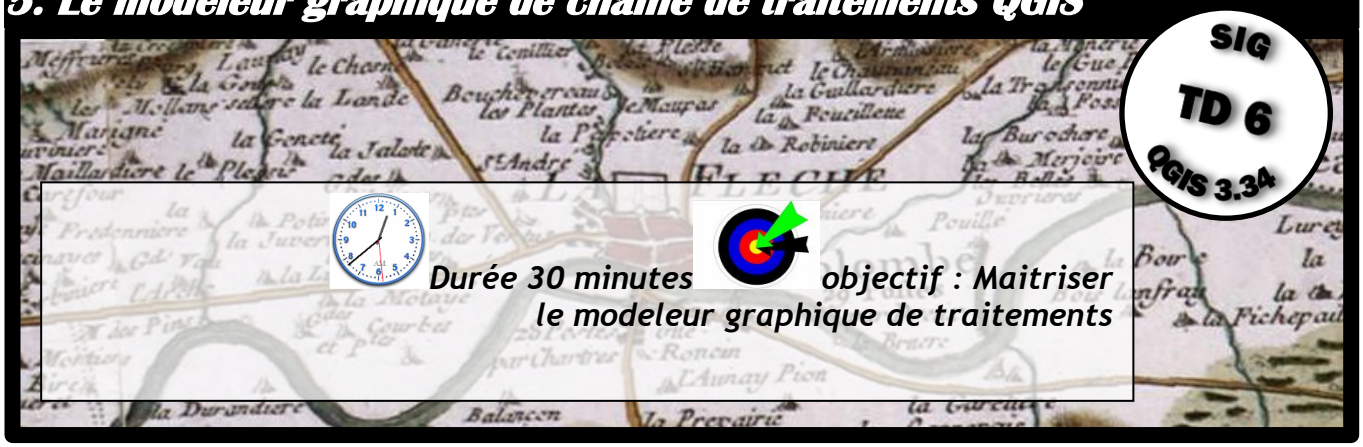
Partie 4. Mode Batch de QGIS

- Le mode Batch permet d'appliquer le même traitement à toute une série de données, Par exemple des centaines de couches dans un répertoire

- Le mode Batch est une fonctionnalité très simple à utiliser

- Avant de penser à faire des scripts penser au mode Batch

5. Le modeleur graphique de chaine de traitements QGIS



Le modeleur graphique de chaine de traitements QGIS, ou GM (Graphic Modeler) est un outil qui permet de développer des « macros » de séquence de traitements **sans coder**, en utilisant un langage iconique et une interface graphique.

Faire un modèle qui calcule la surface totale de « parcelles » par « zones » définie dans une autre couche : exemple le nombre d'hectares de parcelles agricoles RPG par zone IRIS

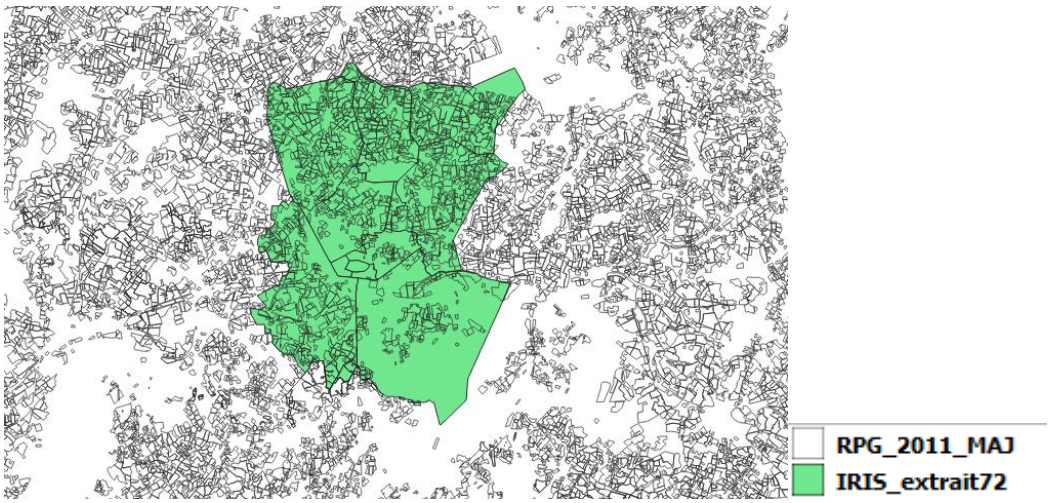
Le registre parcellaire graphique (RPG) contient les parcelles agricoles des déclarations PAC des agriculteurs.

■ Affichez les 2 couches dont on va se servir

LA_FLECHE → Contours_Iris → Carto → IRIS_extrait72

Le champ « Nom_Iris » donne le nom identifiant de la zone iris

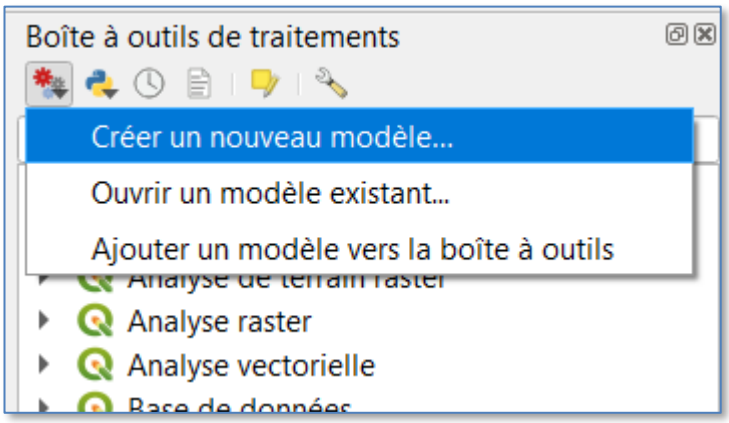
LA_FLECHE → RPG_2011 → RPG_2011_MAJ



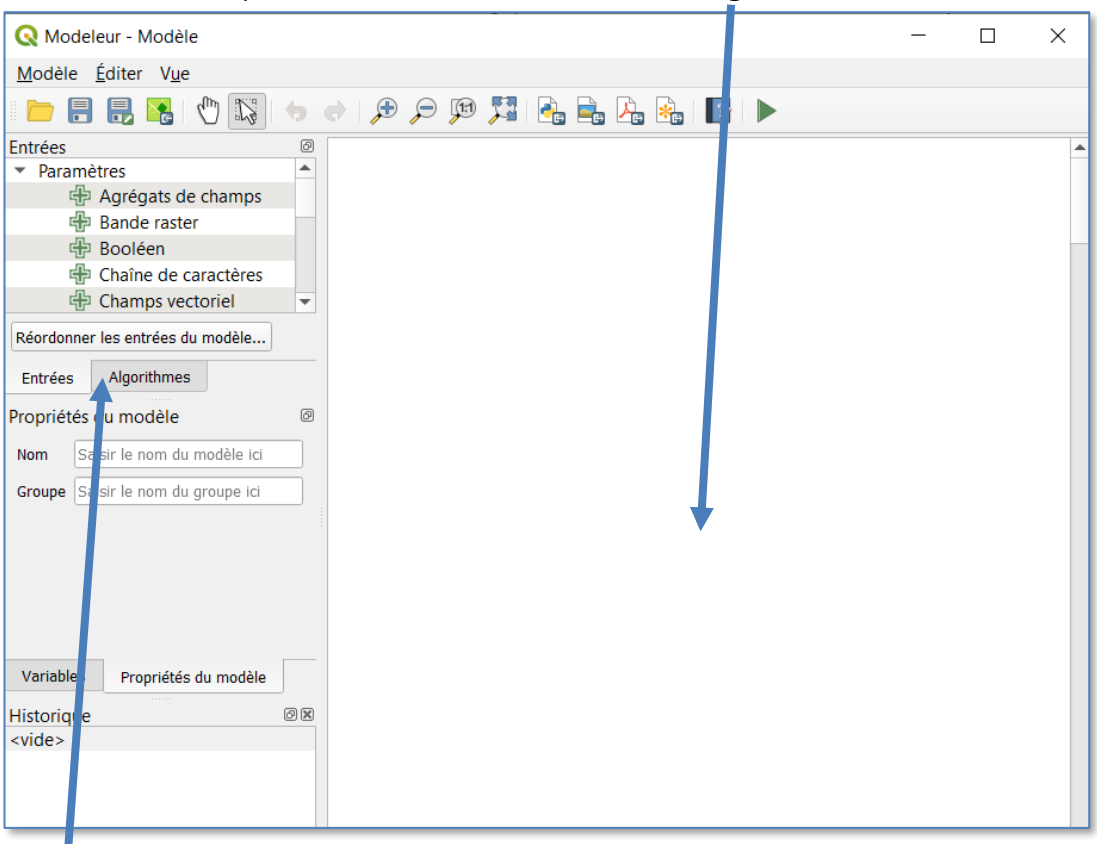
5.1. L'interface du modeleur graphique

■ Ouvrir l'interface du modeleur de traitement

Depuis la boite de traitements, cliquez sur  et choisir « Créer un nouveau modèle »



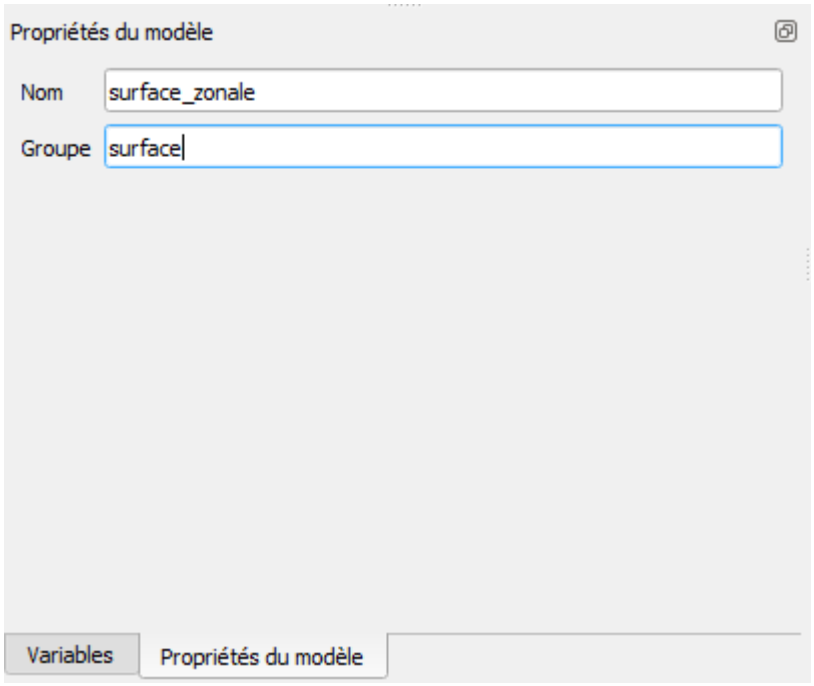
La fenêtre du Modeleur apparait :
Les entrées possibles
ou
Les traitements possibles le futur diagramme de traitements



2 onglets : Choisir affichage des entrées ou des traitements (Algorithmes)

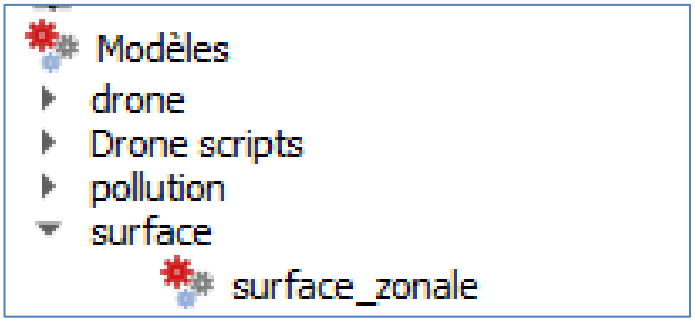
■ Nommer le modèle

Dans les propriétés du Modèle Nom : surface_zonale



■ Enregistrement du modèle avec

Pensez à enregistrer régulièrement au cours de la construction du modèle pour ne rien perdre. Regarder dans la boite à outils QGIS un nouvel outil « surface_zonale » est apparu dans les modèles de la boite à outils de traitements.

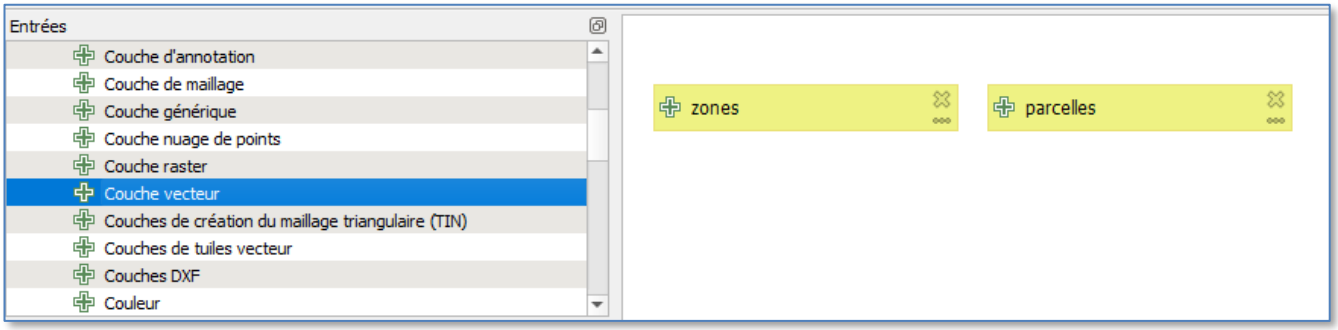


5.2 Ajout des données : paramétrisation des entrées

■ Ajoutez les 2 entrées de type « couche vecteur » et les nommer « zones » et « parcelles »de notre modèle de traitement

Sur l'onglet « Entrées » à gauche
Cliquez sur Couche vecteur pour ajouter nos 2 entrées

En effet nos entrées sont de type couche SIG vecteur



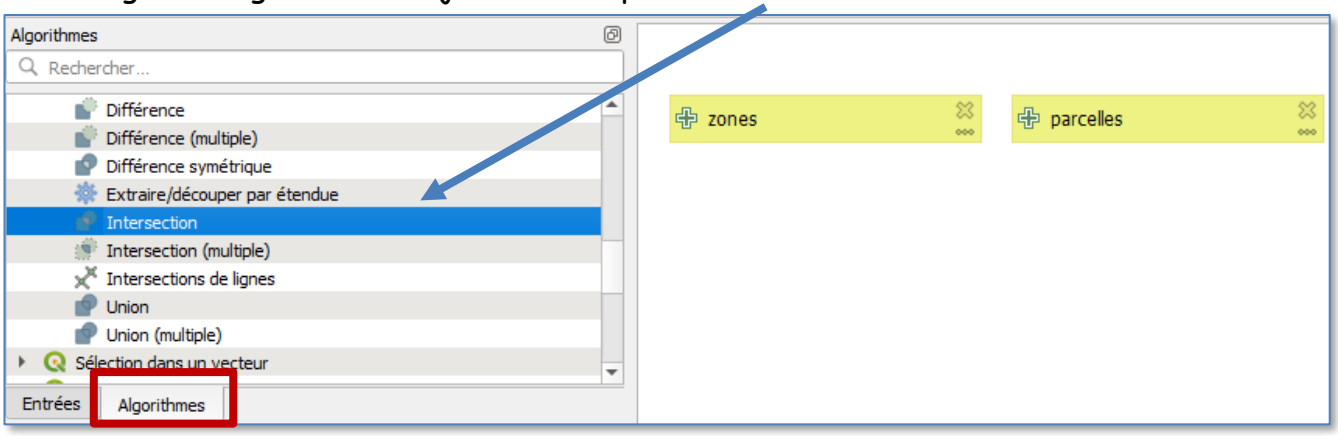
zones et parcelles sont des paramètres d'entrée (variables) ils seront affectés à des données par l'utilisateur au moment de l'exécution, typiquement l'utilisateur choisit dans une liste des couches du projet.

5.3 Ajout des algorithmes (traitements)

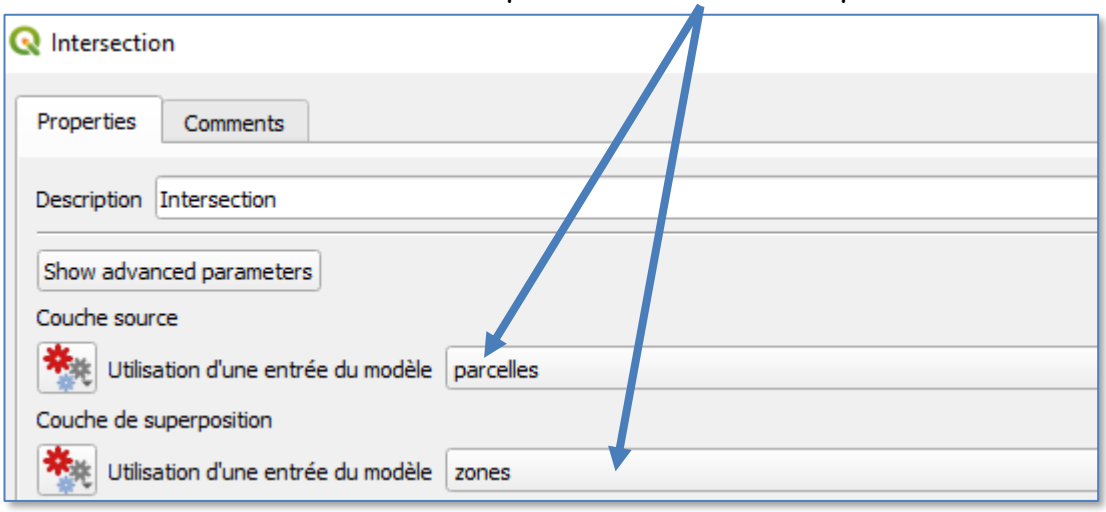
■ Intersecter les parcelles par les zones IRIS : croisement de couches

Recouvrement de vecteur → Intersection

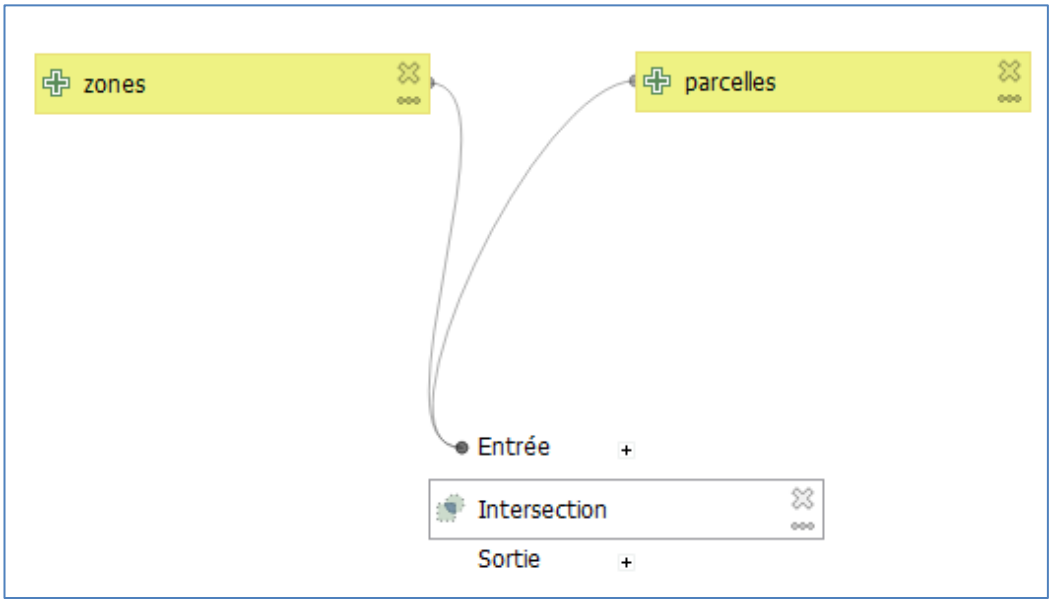
Dans l'onglet « Algorithme » ajouter une opération d'intersection



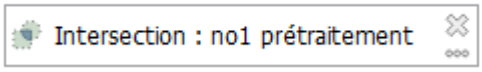
Paramétrer l'intersection avec les paramètres d'entrée : parcelles et zones



Une boite est apparue sur le diagramme et elle est relié a ses paramètres d'entrée



Par défaut le nom de la « boite » est le nom de la fonctions, mais on peut le changer, pour le rendre plus explicite



Surtout pour les gros modèles et pour distinguer les étapes qui sont faites avec la même fonction.

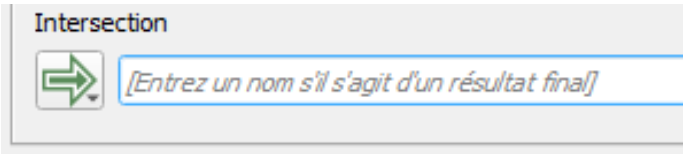
■ Testez l'exécution de votre modèle dans le modeleur

Pour faire cela, Changez temporairement le statut de la sortie, on va lui donner un nom plutôt que de les laisser en fichier temporaire (pour qu'il se charge en mémoire).

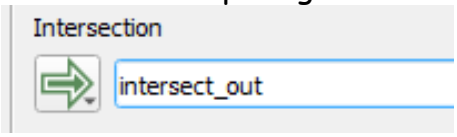
sur 

et Editer

En bas de la boite de dialogue, le champ intersection (même nom que la fonction) est le fichier sorti en résultat, par défaut si vide, c'est un résultat intermédiaire, temporaire.

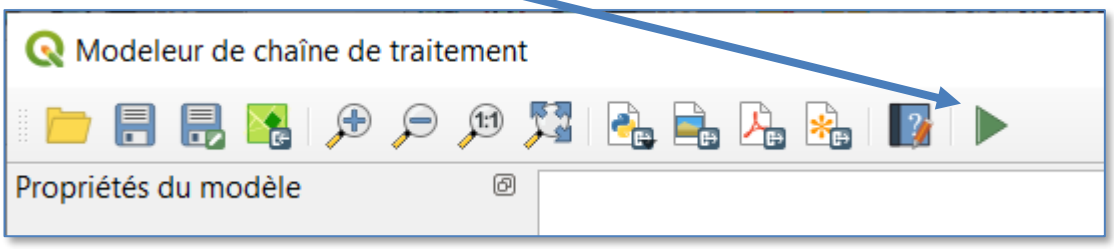


Entrer un nom pour générer un paramètre résultat final : par exemple « intersect_out »

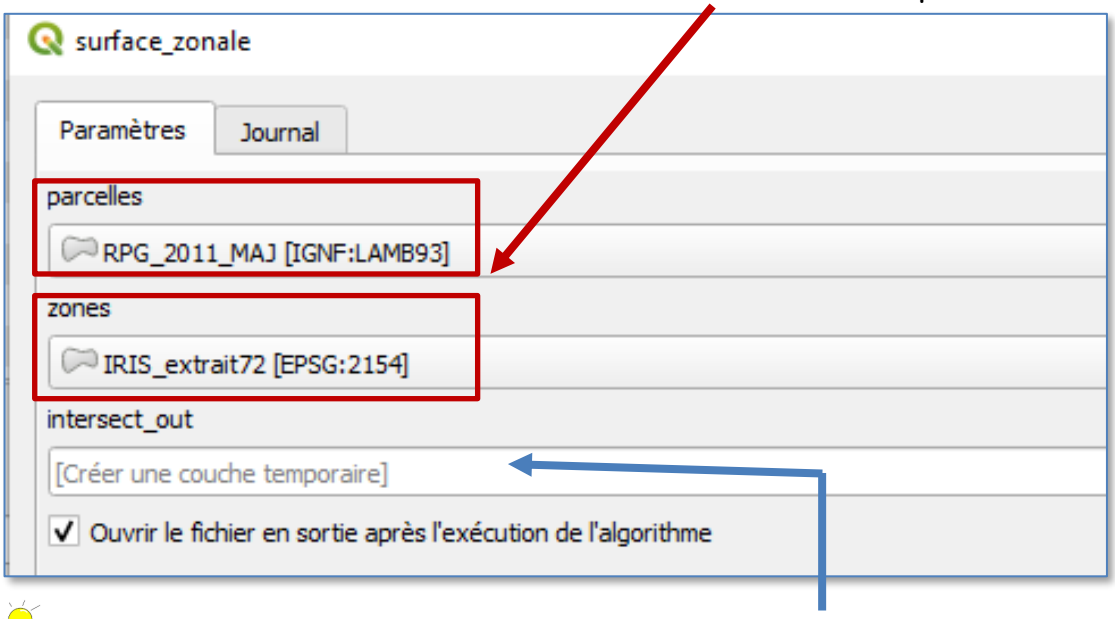


On peut avoir plusieurs paramètres résultats dans un même modèle, si nécessaire.

Testez l'exécution avec 

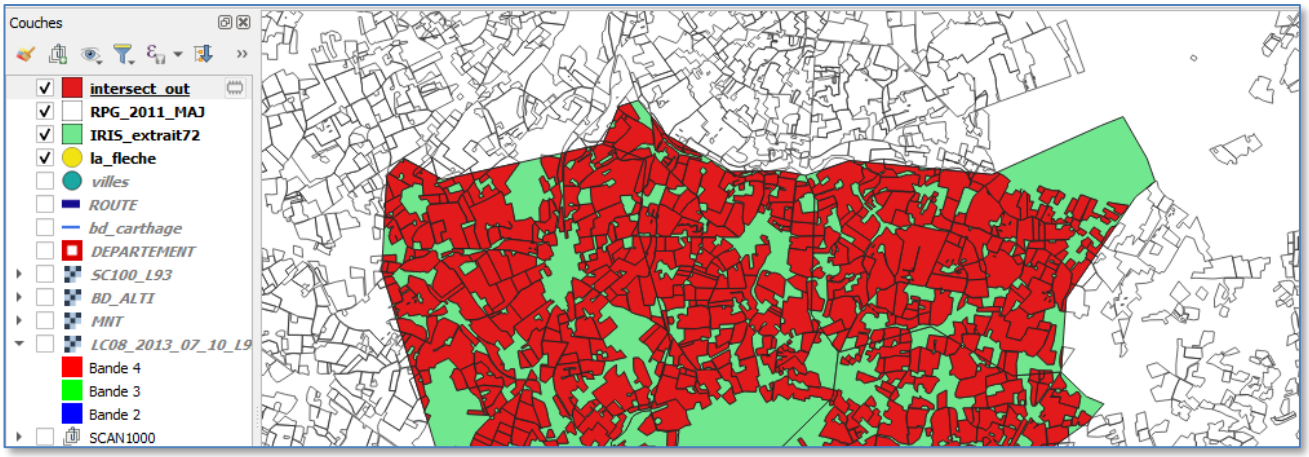


L'utilisateur choisit dans les listes les couches à affecter aux paramètres



 On peut mettre ici un fichier permanent (pas temporaire) si souhaité

Regarder dans le projet QGIS une couche « intersect_out » a été ajoutée

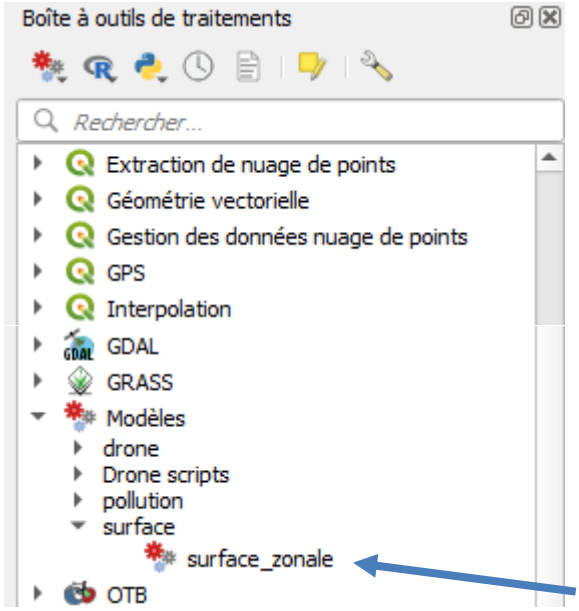


5.4 Le nouveau modèle dans la boite à outils de traitement

■ Testez l'exécution de votre modèle dans la boite à outils de traitements

Enregistrer votre modèle et fermer la fenêtre Modeleur

Depuis la boite à outils de traitements de la fenêtre principale QGIS



Double cliquer sur le modèle « surface_zonale »

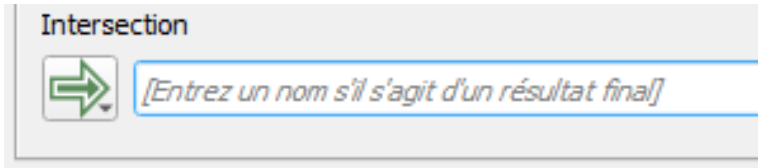
L'exécution et le résultat sont identiques au point précédent, notre outil est devenu un outil à part entière de la boite de traitements, il pourra même être intégré à d'autres modèles ou scripts.

■ Ouvrir de nouveau la fenêtre du modeleur par sur « surface_zonale » dans la boite de traitements et choisir « éditer le modèle »

5.5 Connecter les algorithmes entre eux : input <-> output

■ Calculer la surface des parcelles intersectées

Supprimer la sortie précédente de l'intersection pour revenir à

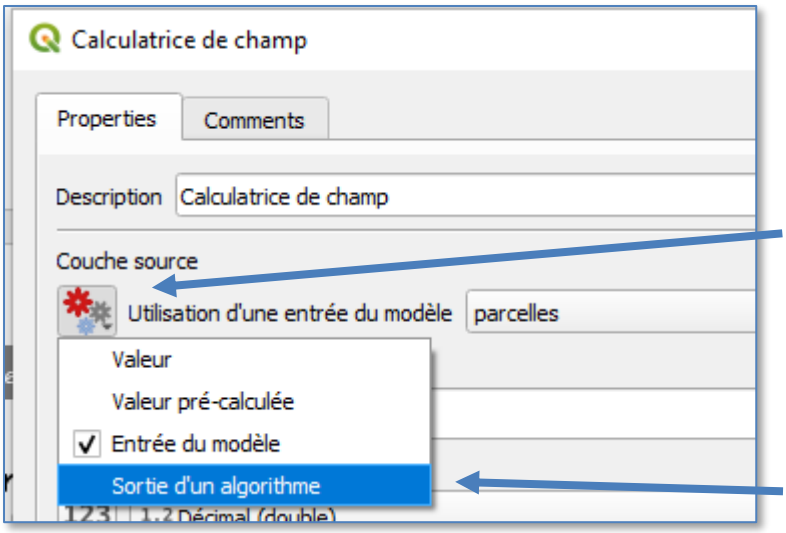


■ Ajouter au modèle une calculatrice de champ:

Table vecteur → Calculatrice de champ

Créer un champ « SURFACE » où on calculera la surface en hectares ($area / 10000)

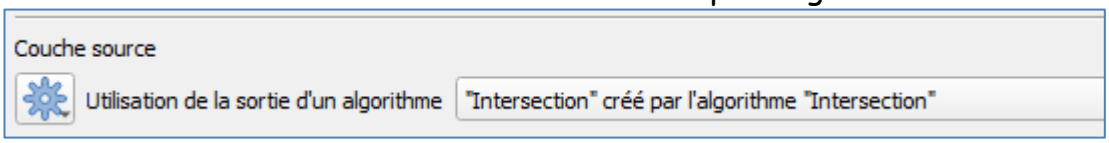
On va connecter la sortie de l'intersection (couche temporaire) avec l'entrée de la calculatrice.



Cliquer sur le bouton

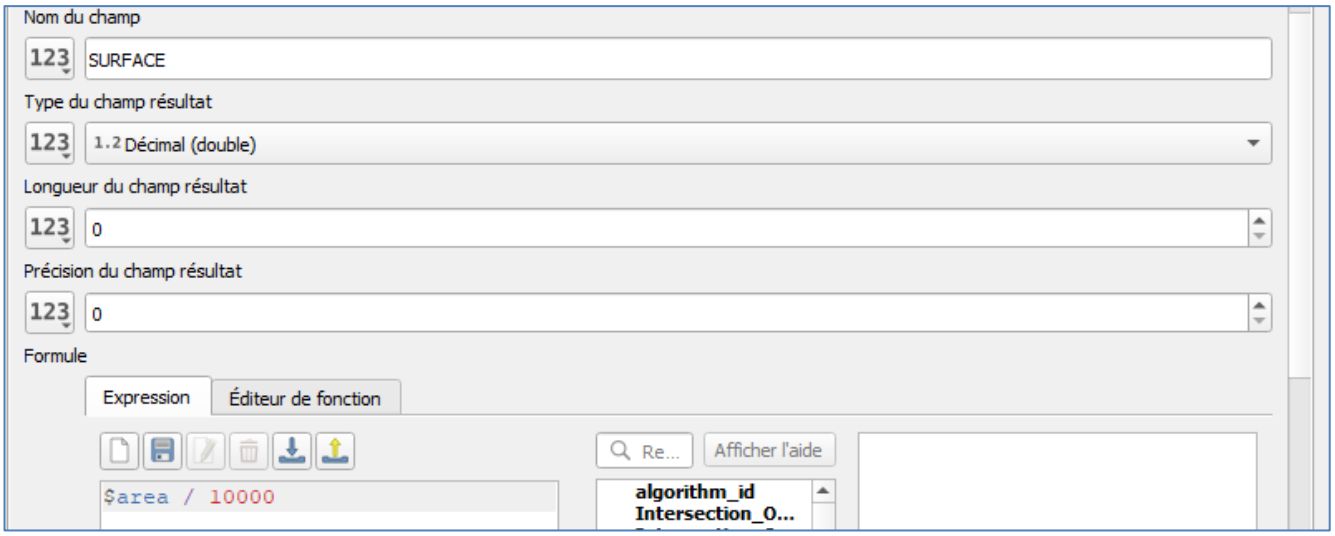
Choisir comme type d'entrée : Sortie d'un algorithme précédent

L'icone du bouton change, montrant qu'on a changé de type de paramètre
Et on choisit dans la liste « intersection » créé par l'algo. Intersection

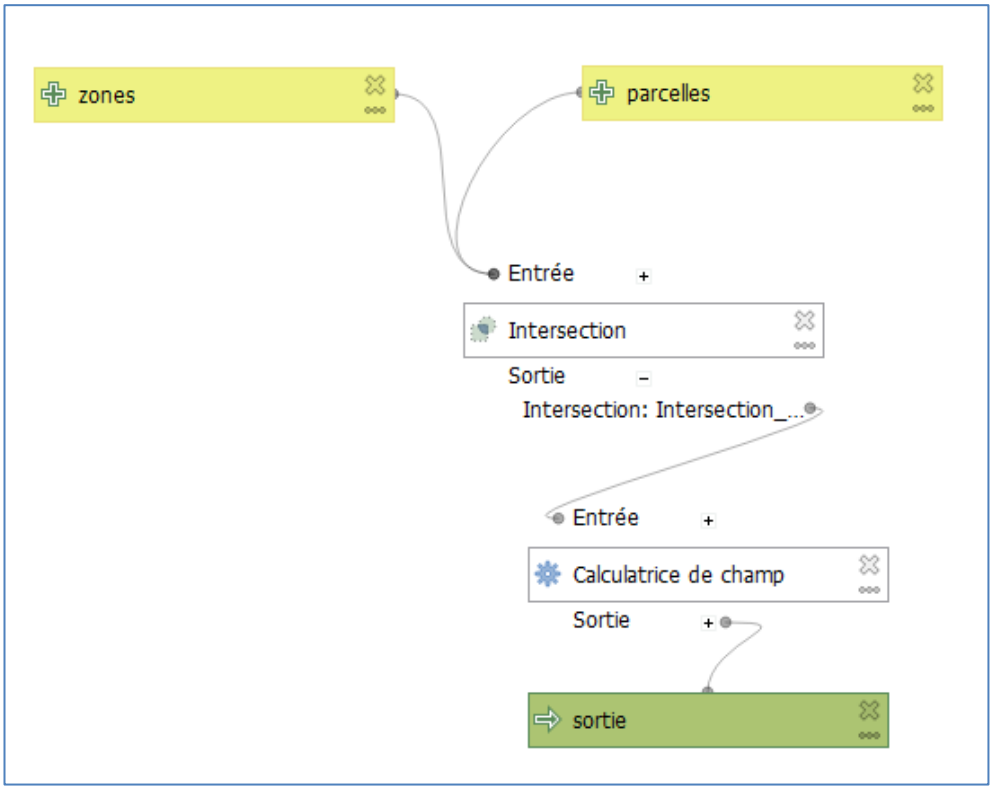


Ce processus est très important car c'est qui va permettre d'enchainer les étapes du modèle les unes à la suite des autres, la sortie de la première étape est ainsi automatiquement connectée sur l'entrée de la seconde etc …

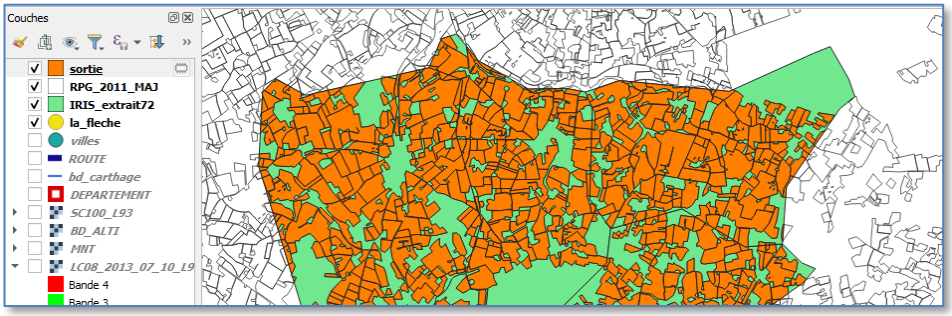
Dans la suite de la boite de dialogue : Créer un champ « SURFACE » de type Décimal et ajouter une formule pour y calculer la surface des polygones

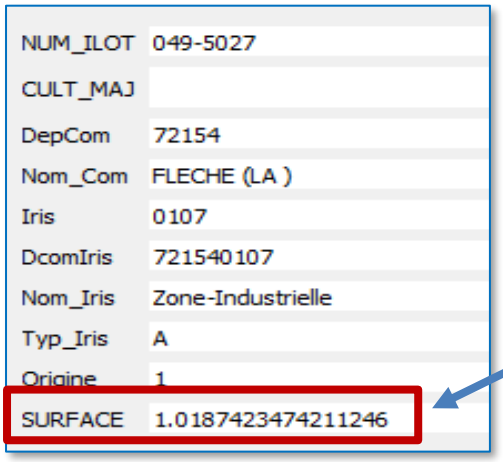


Nommer le résultat de la calculatrice « sortie » pour voir le résultat dans QGIS



Géographiquement le résultat est identique à l'étape d'intersection





Mais si on ouvre la table attributaire

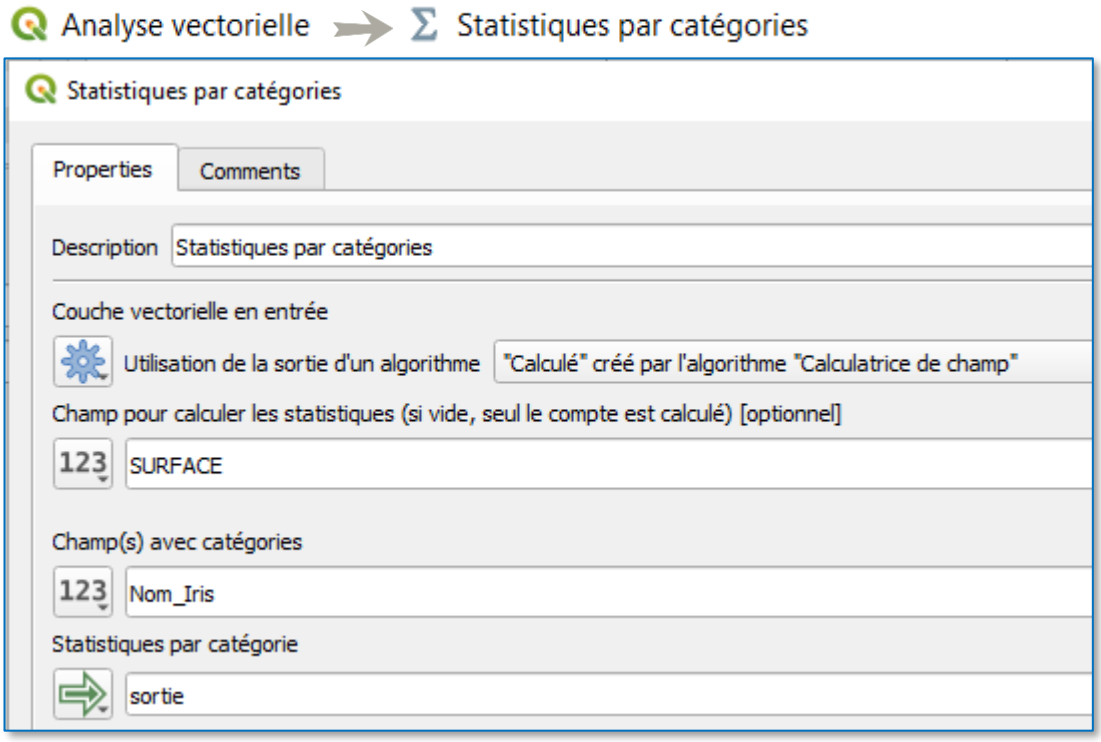
Un nouveau champ « SURFACE » est apparu

■ Récapituler les surfaces précédentes par IRIS

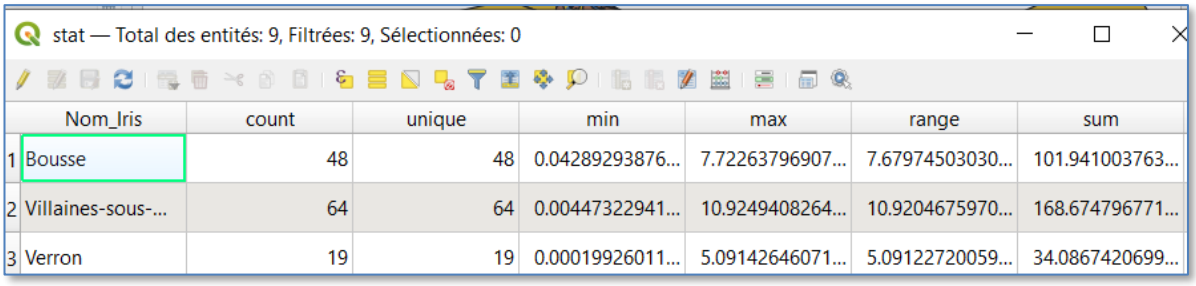
De manière à calculer la surface totale de parcelles pour chaque zone IRIS

Supprimer le paramètre « sortie » de la calculatrice

Ajouter « statistiques par catégorie » à votre modèle



Regarder la table résultat « sortie » qui donne les statistiques de surface par IRIS, en particulier le champ « sum » donne la surface totale de parcelles à moins de 150m d'une rivière, en hectares.


stat — Total des entités: 9, Filtrées: 9, Sélectionnées: 0

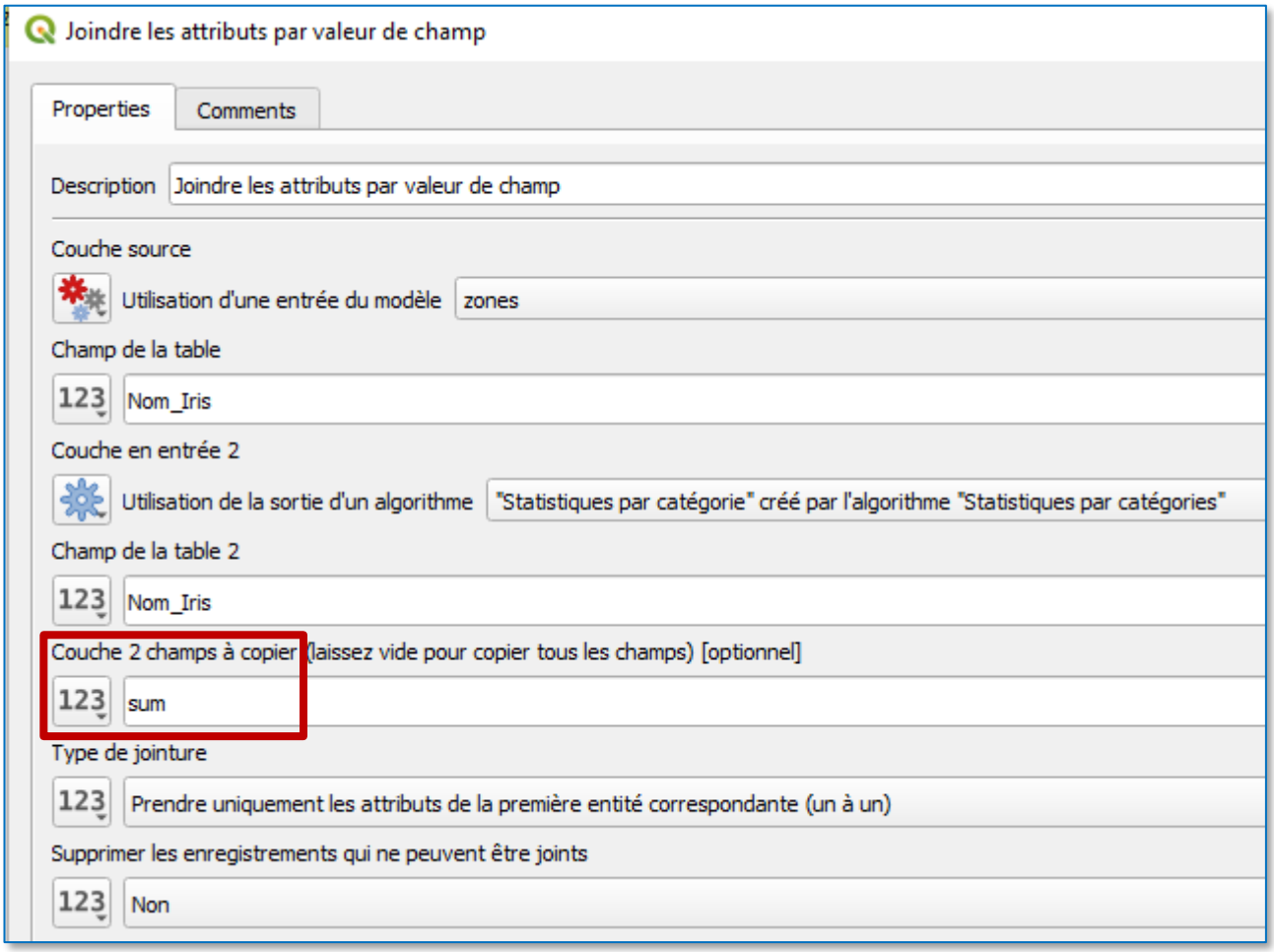
	Nom_Iris	count	unique	min	max	range	sum
1	Bousse	48	48	0.04289293876...	7.72263796907...	7.67974503030...	101.941003763...
2	Villaines-sous-...	64	64	0.00447322941...	10.9249408264...	10.9204675970...	168.674796771...
3	Verron	19	19	0.00019926011...	5.09142646071...	5.09122720059...	34.0867420699...

■ Joindre la table précédente sur les IRIS

Supprimer le paramètre « sortie » de l'étape précédente
Ajouter au modèle une opération : « joindre les attributs par valeur de champ »

Outils généraux pour les vecteurs → Joindre les attributs par valeur de champ

Attention au sens de la jointure.

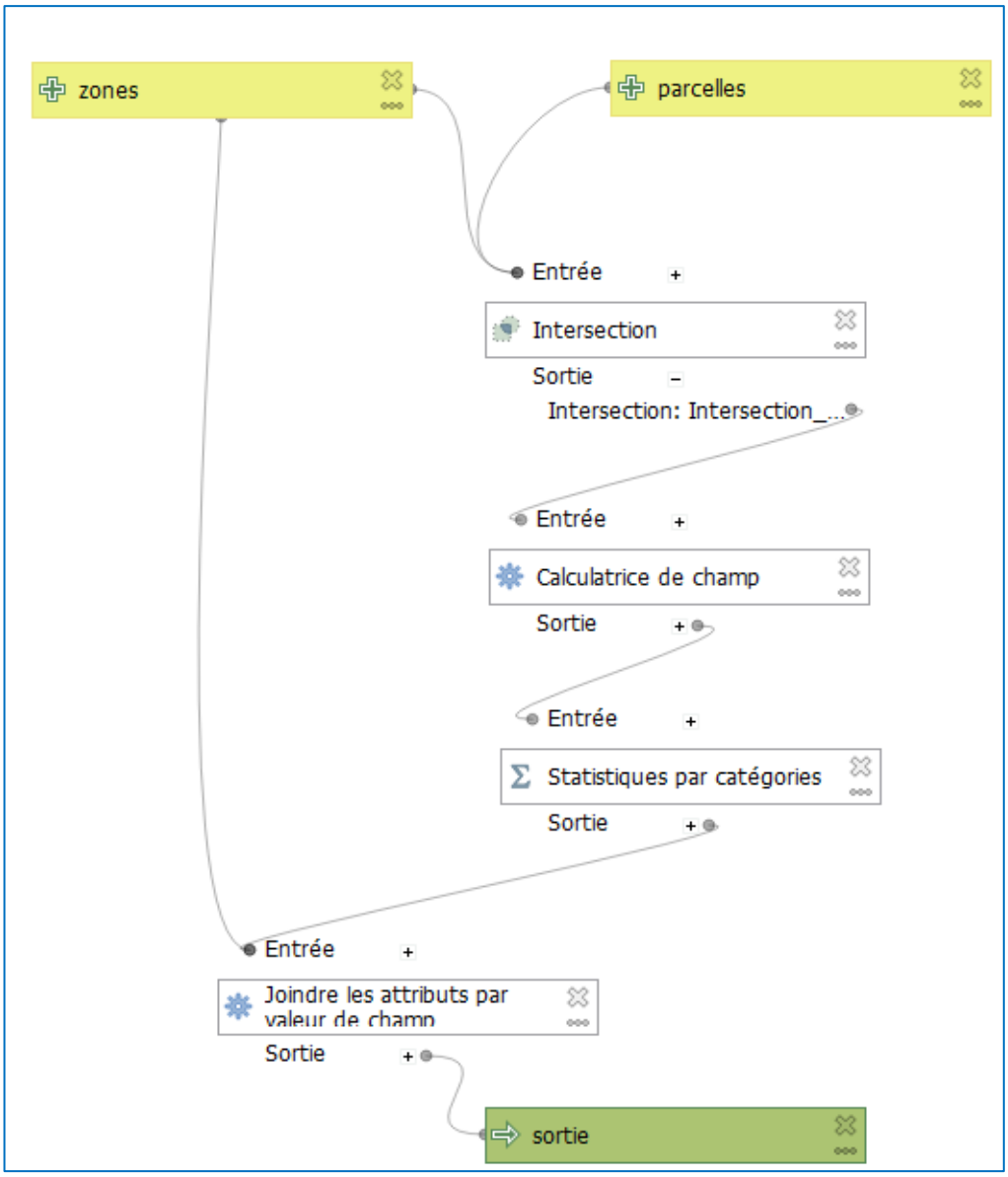

Joindre les attributs par valeur de champ

Properties | Comments

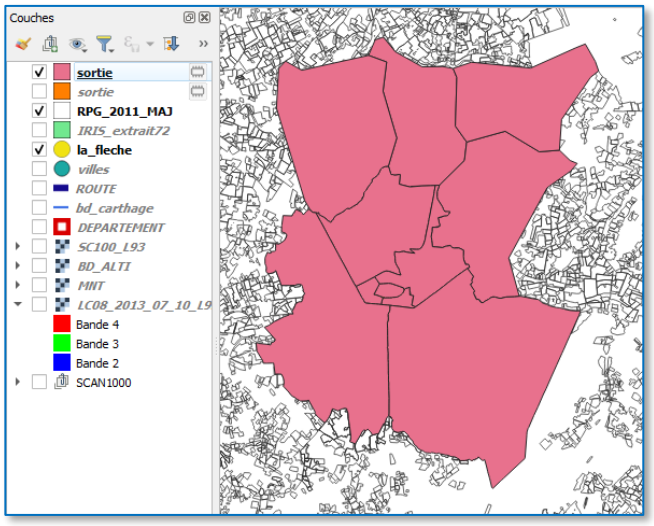
Description: Joindre les attributs par valeur de champ

Couche source
Utilisation d'une entrée du modèle: zones
Champ de la table
Nom_Iris
Couche en entrée 2
Utilisation de la sortie d'un algorithme: "Statistiques par catégorie" créé par l'algorithme "Statistiques par catégories"
Champ de la table 2
Nom_Iris
Couche 2 champs à copier (laissez vide pour copier tous les champs) [optionnel]
sum
Type de jointure
Prendre uniquement les attributs de la première entité correspondante (un à un)
Supprimer les enregistrements qui ne peuvent être joints
Non

Pour éviter d'alourdir la table des Iris on ne demande la jointure que du champ « sum » qui contient la surface totale des parcelles du RPG (par défaut tous les champs seraient copiés)



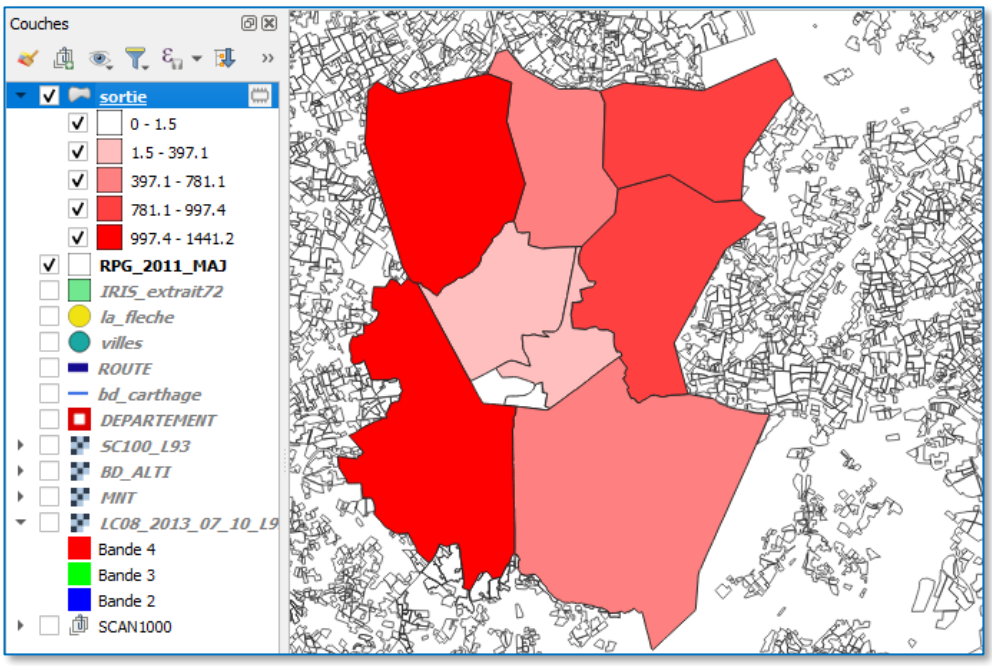
Exécuter  on voit que la sortie a maintenant la même géométrie que les zones Iris



Regarder les attributs de « sortie »
Sum a été importé par la jointure

DepCom	72044
Nom_Com	BOUSSE
Iris	0000
DcomIris	720440000
Nom_Iris	Bousse
Typ_Iris	Z
Origine	NULL
sum	781.1083839535393

Dans la fenêtre QGIS faire une symbologie par dégradés de couleurs sur « sum »

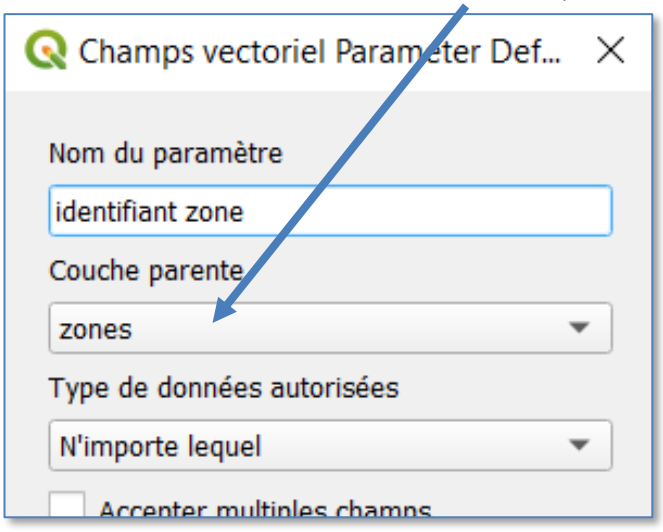


5.6 Paramétrer un champ

■ Paramétrer le champ identifiant des zones

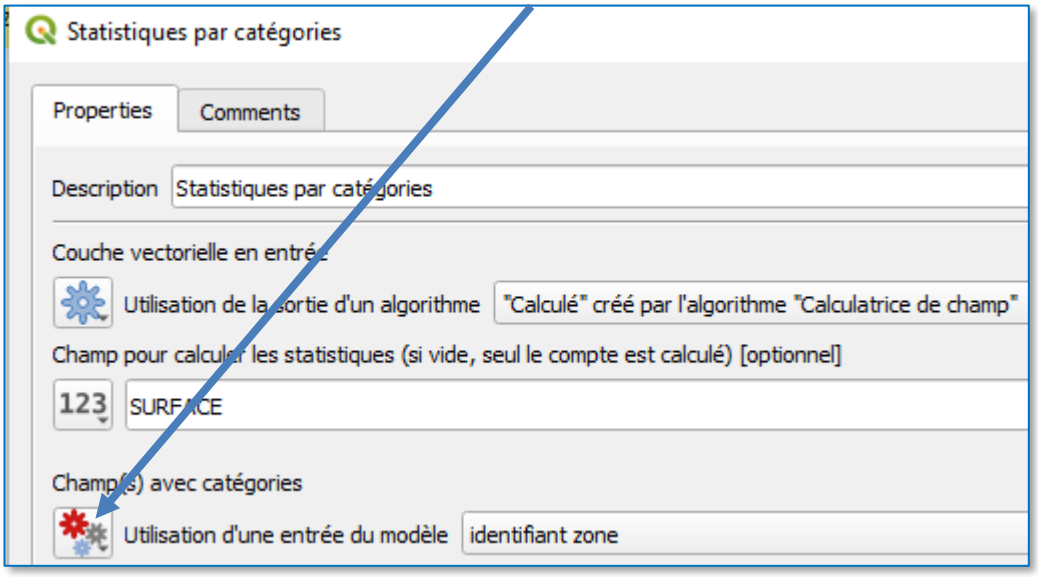
Pour pouvoir utiliser ce modèle avec d'autres couches de zone que les IRIS il faut le rendre indépendant du champ identifiant des zones « Nom_Iris » on va donc **paramétrer ce champ dans le modèle**.

Ajoutez un paramètre de type Champs vectoriel
Appelez le « identifiant_zone »
Le lier à la couche d'entrée « zones » (couche parente)



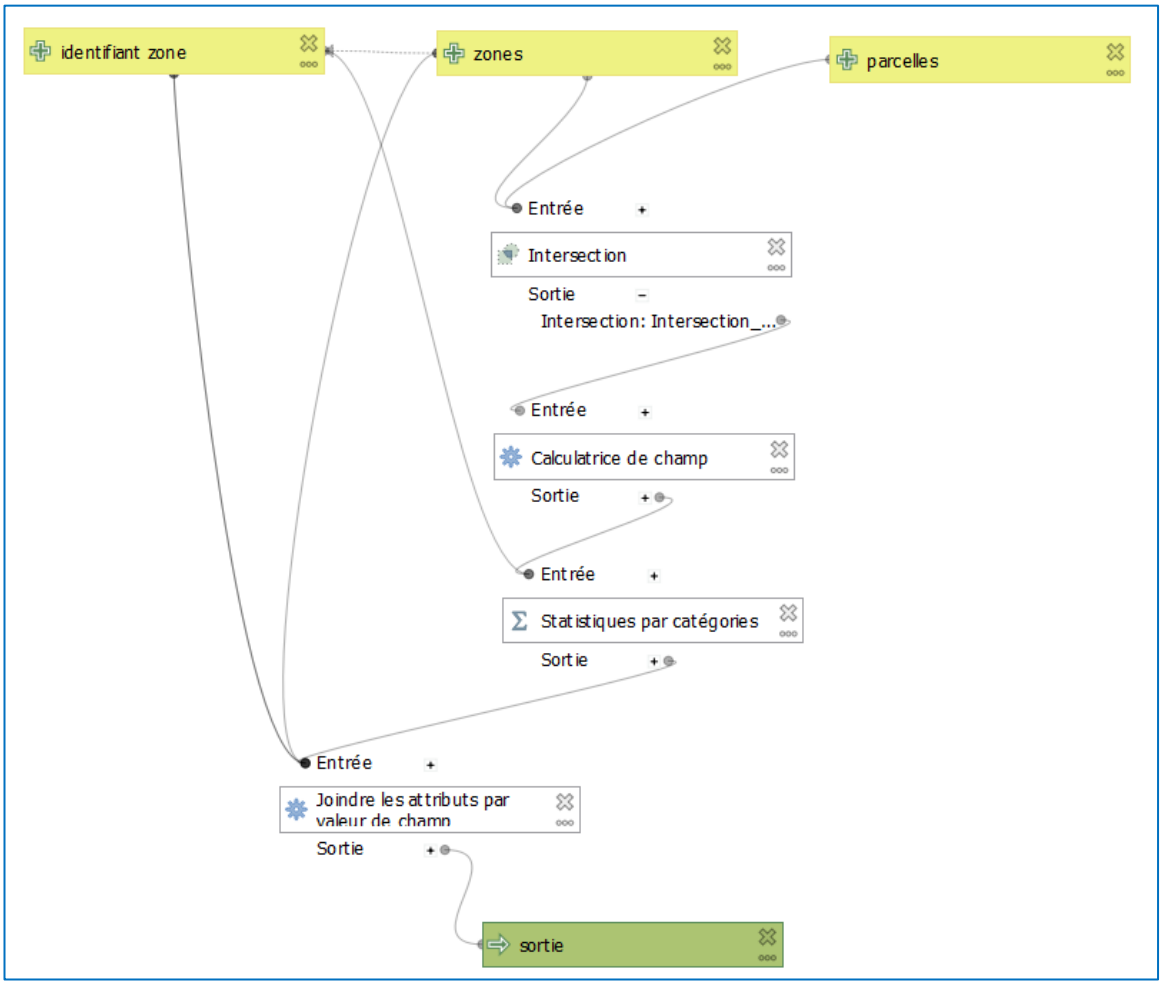
Relier ce champ aux traitements ***Statistiques par catégories* et jointure**
Testez votre modèle avec ce changement

Pour statistiques par catégories changer le type pour « entrée du modèle »



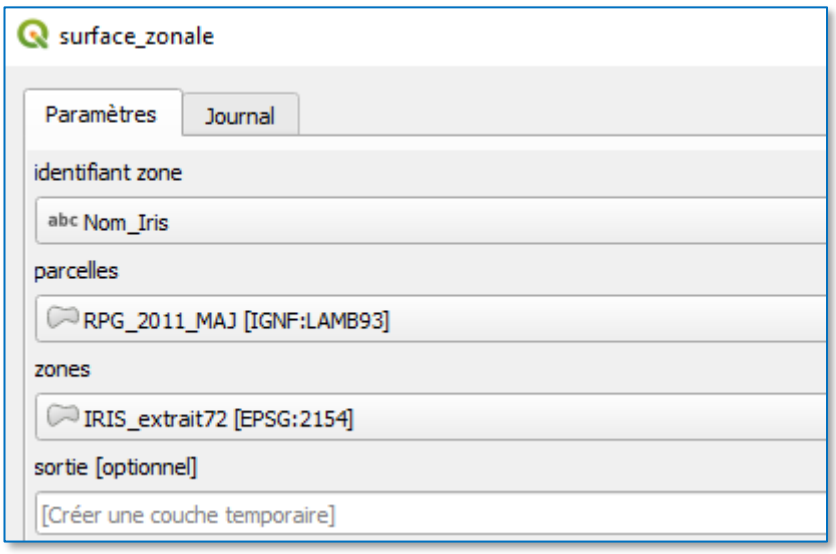
Puis choisir « identifiant zone »

Faire la même chose pour la jointure



■ Appliquer votre modèle sur IRIS

Il faut maintenant préciser le champ qui sert d'identifiant pour les zones IRIS

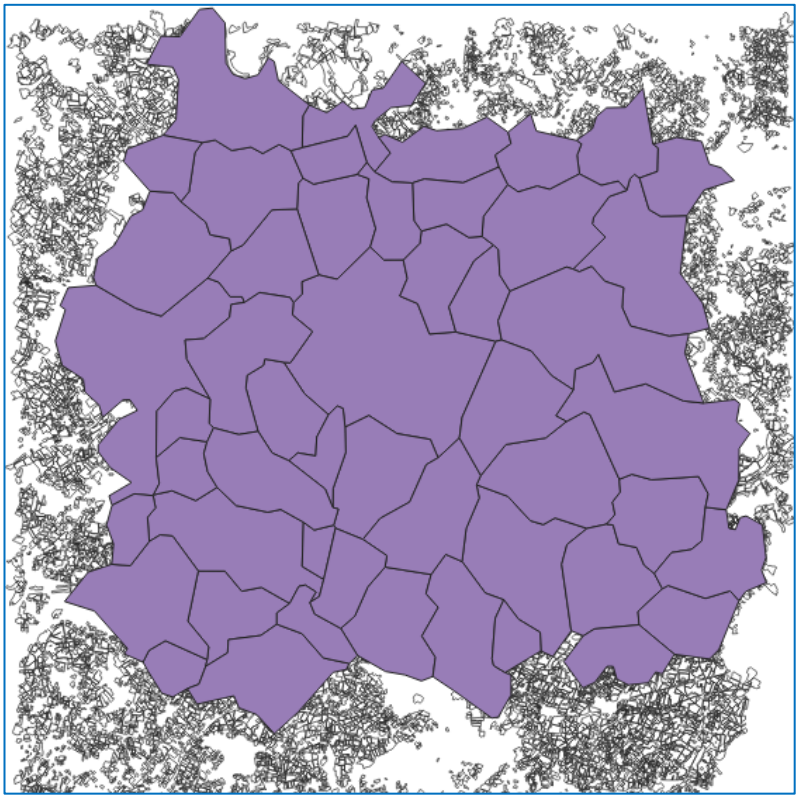


■ Appliquer votre modèle sur une autre couche de zone que les IRIS

Afficher « COMMUNES_RPG »

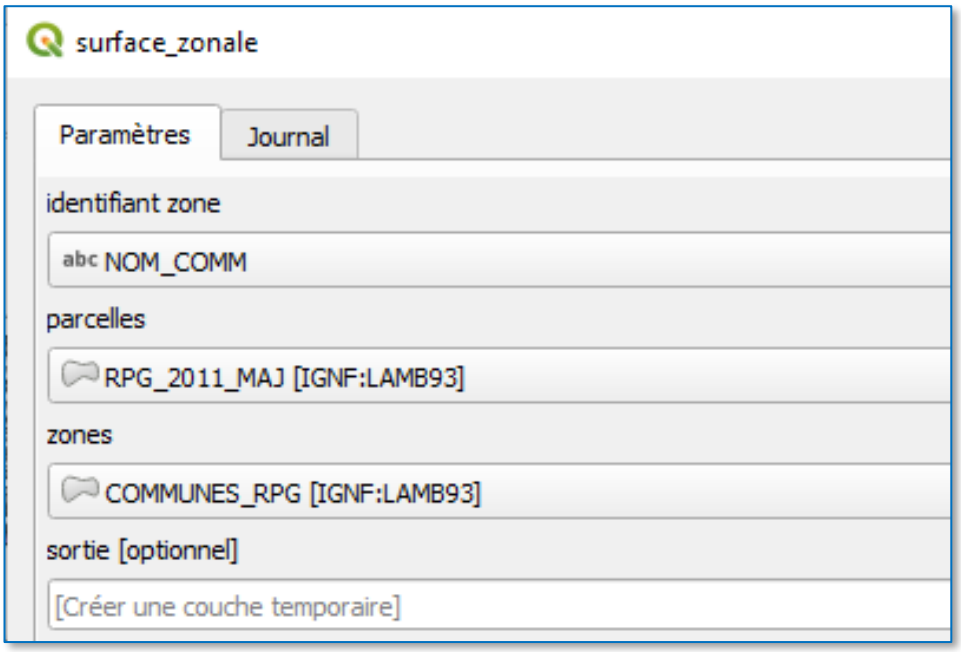
LA_FLECHE → RPG_2011 → COMMUNES_RPG

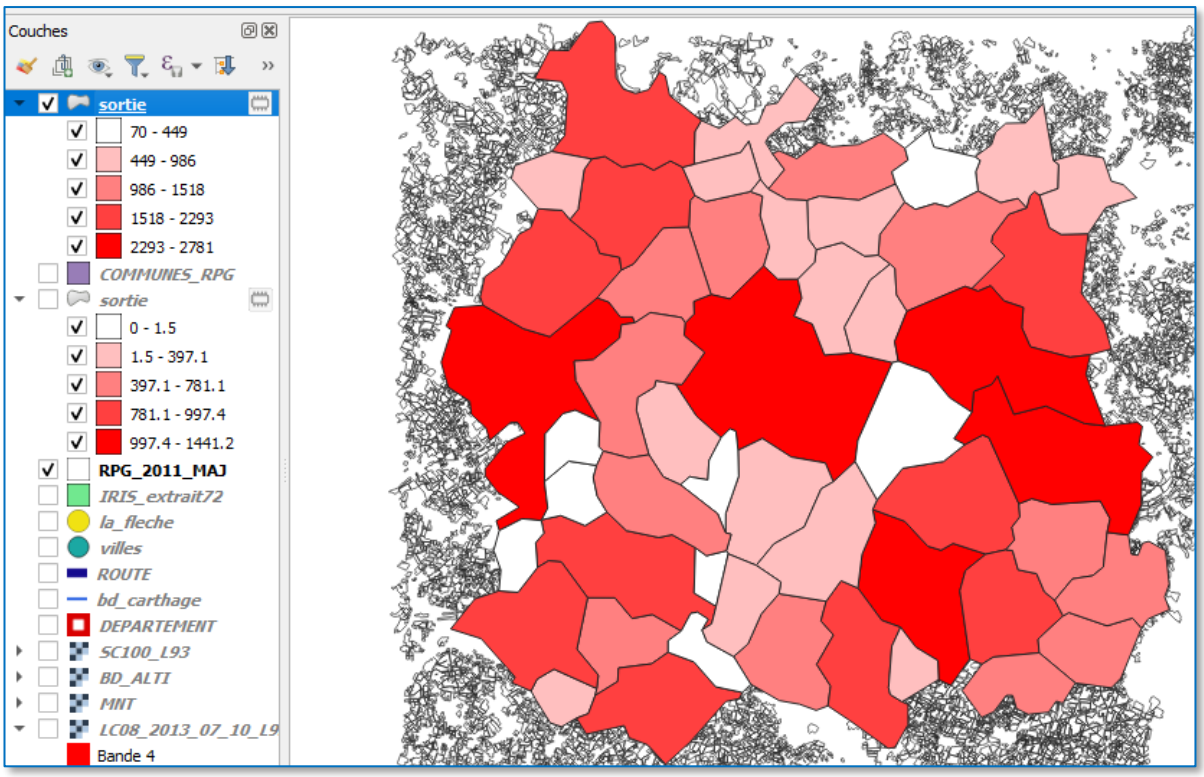
Ce sont les communes où on a l'information des parcelles RPG



Lancer votre modèle en utilisant COMMUNES_RPG comme couche de zones

Le champ « NOM_COMM » contient le nom de la commune et pourra être utilisé comme identifiant de zone.





À retenir :

Partie 5. Le modeleur graphique de traitements

- Le modeleur graphique permet de faire des « macros » sans coder avec un langage informatique.

- Il est assez simple à utiliser

- Il permet de standardiser une chaine de traitement : elle sera toujours exécutée avec les mêmes outils et les mêmes paramètres, elle sera ainsi indépendante de l'opérateur.

- Si on paramètre bien le modèle on peut l'appliquer à différents jeux de données

- A priori on ne peut pas encore faire des boucles dans les modèles

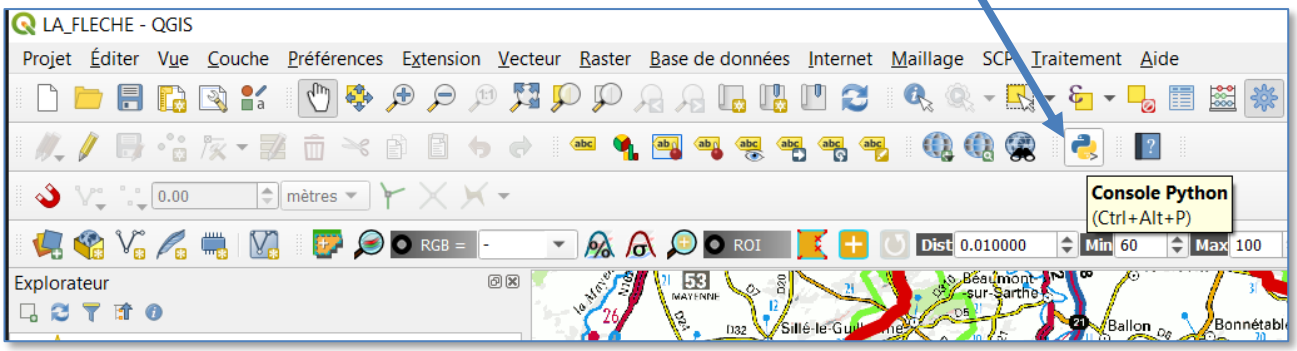
- Il y a la possibilité de faire des IF avec les branches conditionnelles

- Pour des traitements moins séquentiels il faut passer au codage en PYTHON

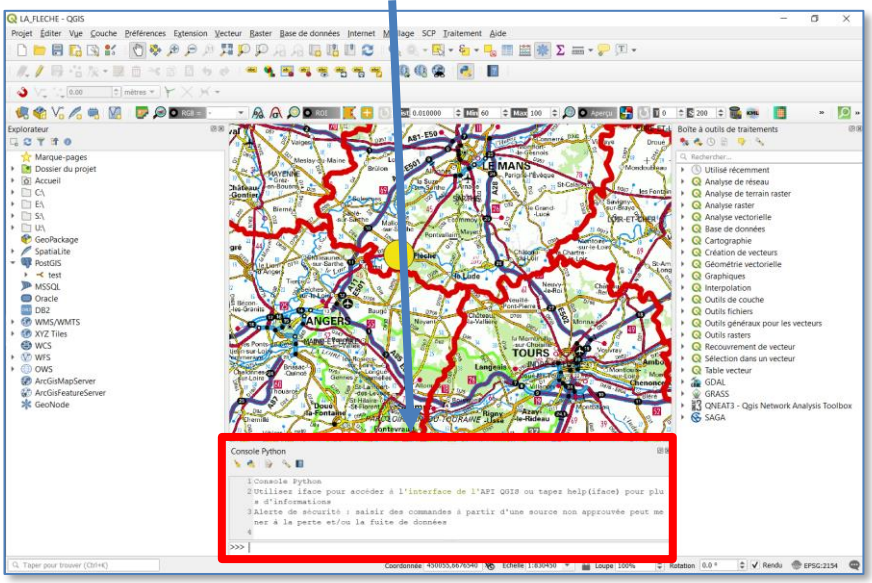
6. Script python et R dans l'interface de QGIS

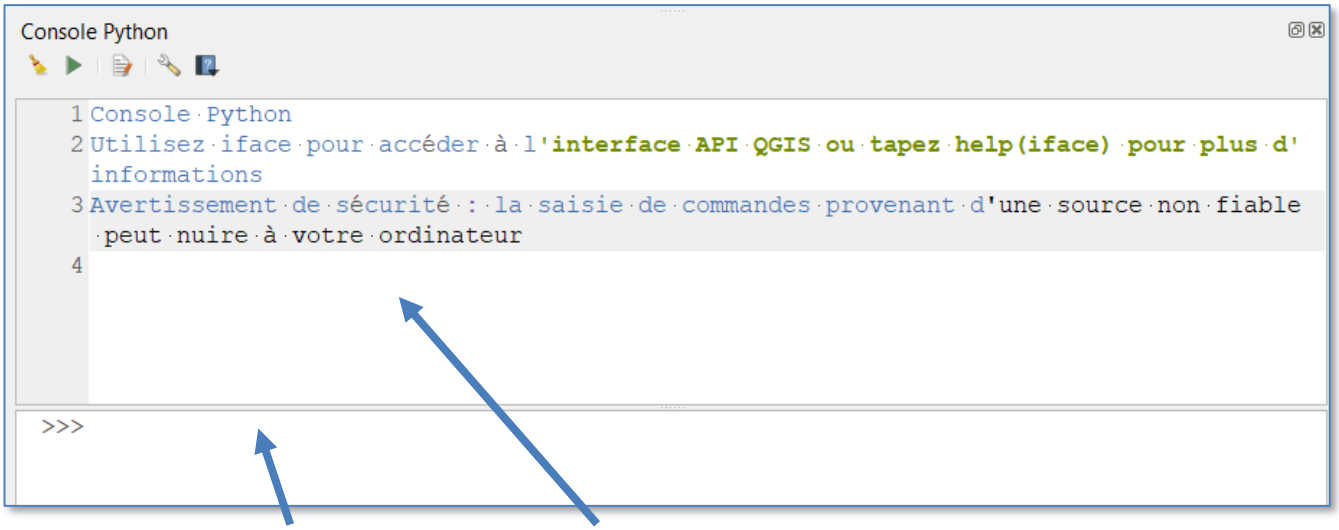
6.1. Code dans la console python de QGIS et l'interface iface

On peut accéder à la console python de QGIS avec le bouton 



La fenêtre de console apparaît





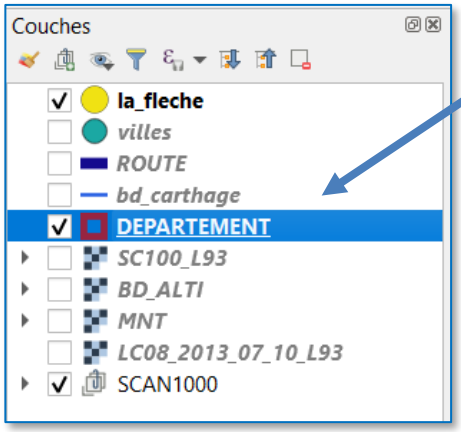
On peut saisir du code Python ici les affichages se font ici

Au démarrage de la console est automatiquement exécuté le code suivant (caché), qui intègre la librairie de base de QGIS

```
from qgis.core import *
import qgis.utils
```

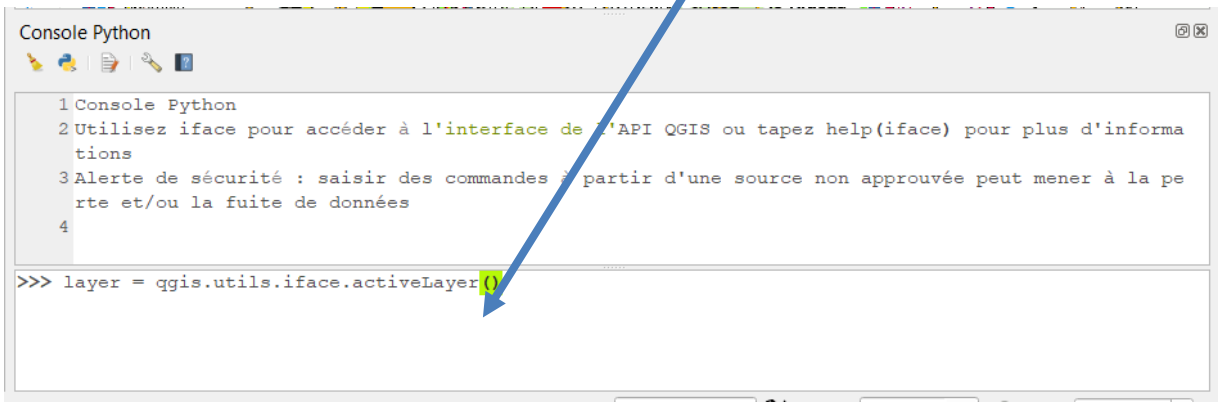
L'interface iface() permet de faciliter l'accès aux objets du projet QGIS.

Compter le nombre d'enregistrement de la couche active dans QGIS

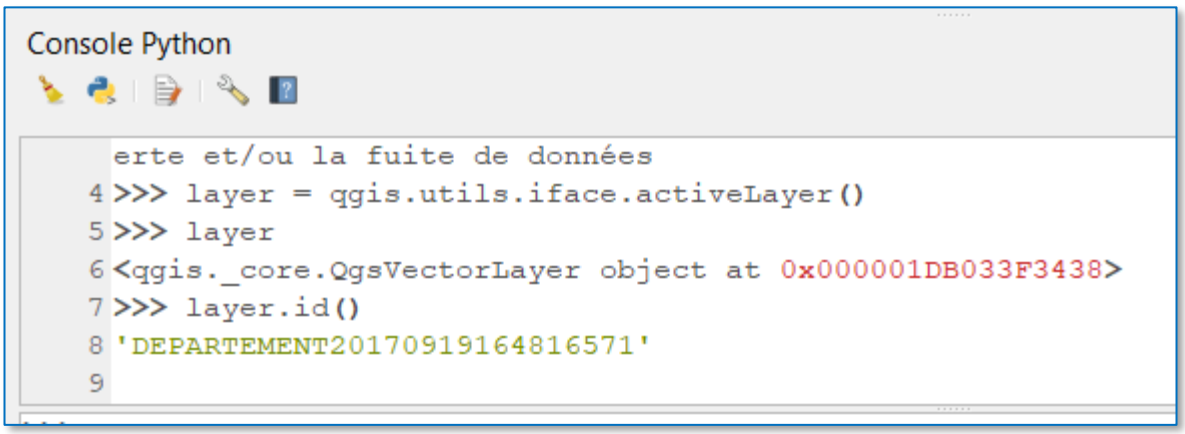


Cliquez dans le volet des couches
sur la couche DEPARTEMENT
Puis dans la console Python saisir le code suivant

```
layer = qgis.utils.iface.activeLayer()
layer
layer.id()
```

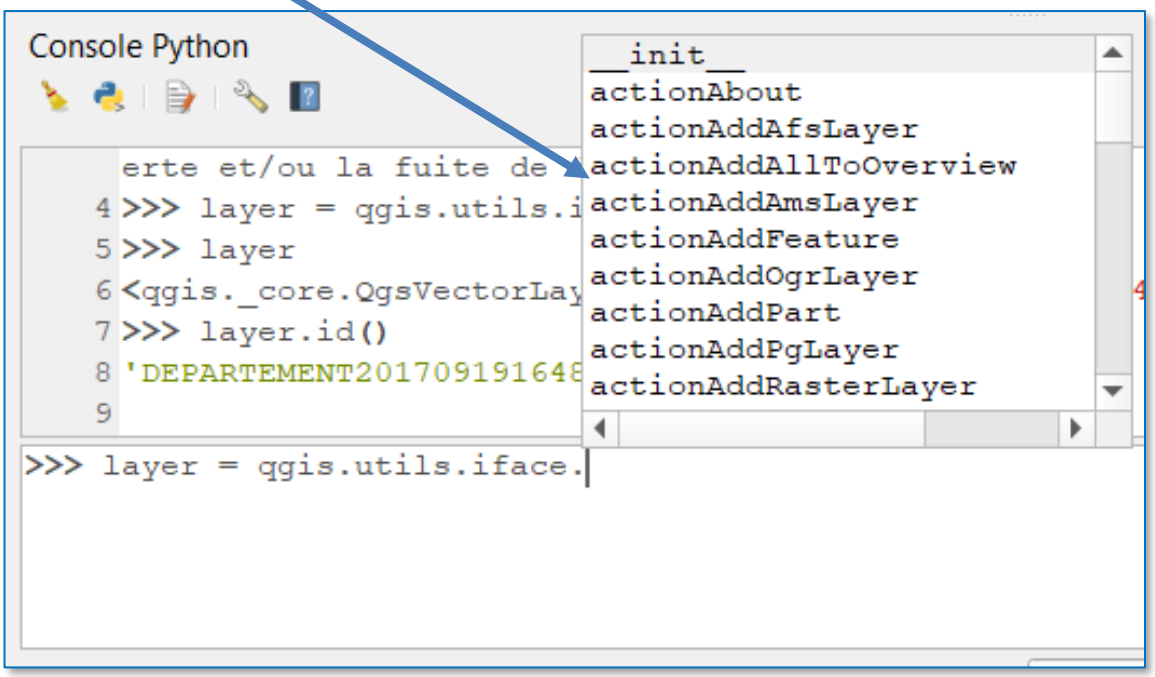


Ce code permet de récupérer un pointeur sur la couche active de la table des matières



Ou *print(layer.id())*

Remarquez que pendant la saisie du code une aide affiche les fonctions en mode auto complétion



Pour compter les enregistrements :

```
layer.featureCount()
```

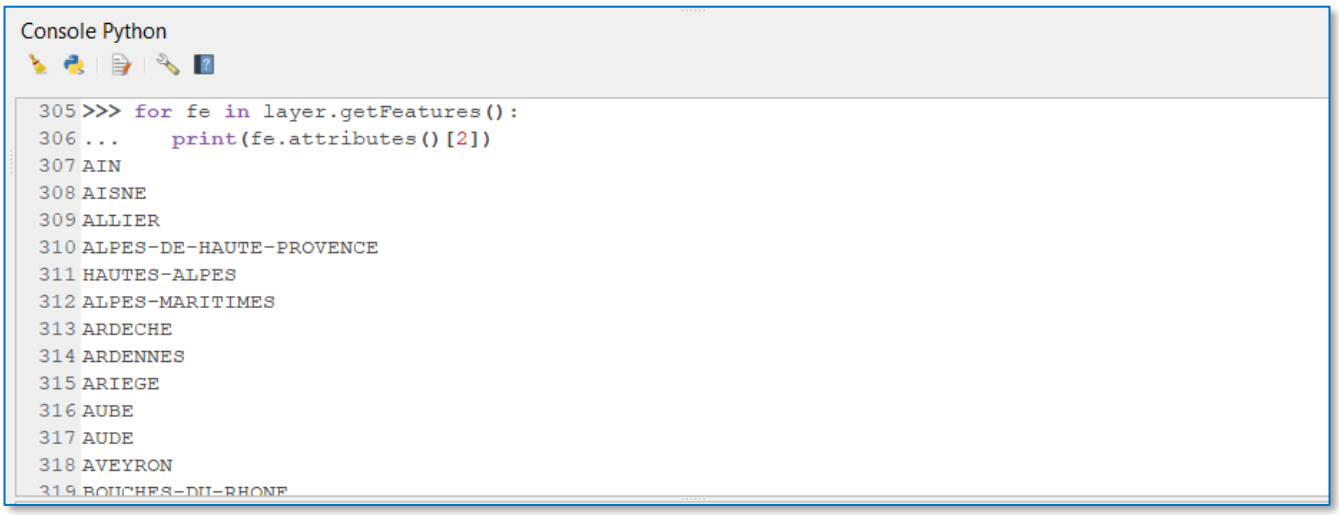
```
>>> layer.id()
'DEPARTEMENT20170919164816571'
>>> layer.featureCount()
96
```

Faire une boucle qui imprime le nom de tous les départements

```
for fe in layer.getFeatures():
    print(fe.attributes()[2])
```

Attention à l'indentation en python (décalage en colonne) c'est ce qui marque le début d'un nouveau bloc d'instructions.

Le print() s'affiche dans la fenêtre de console



Deux fois entrée sur la dernière ligne de code pour lancer l'évaluation

QGIS Python API documentation est sur :

https://qgis.org/pyqgis/3.34/

et aussi

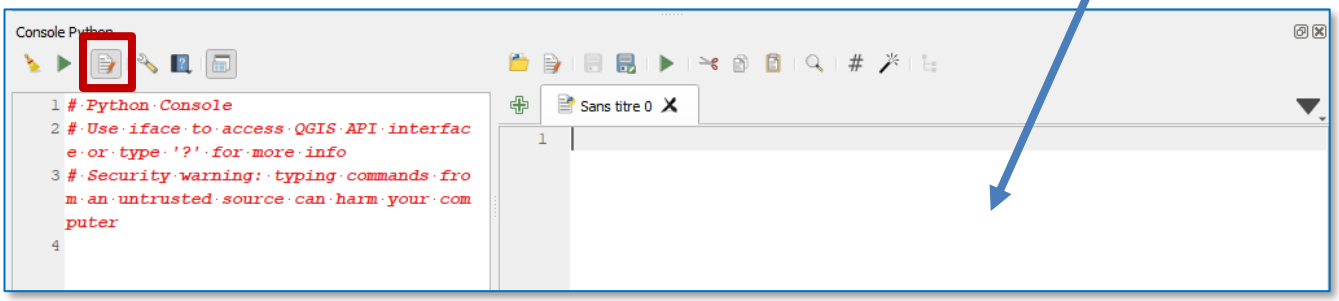
https://docs.qgis.org/3.34/en/docs/

6.2 Exemples avec iface dans un fichier script de la console

6.2.1 Saisir et exécuter un fichier script dans la console

Dans la fenêtre de la console le bouton permet d'ouvrir un éditeur de code

Quand on clique sur ce bouton un éditeur apparait

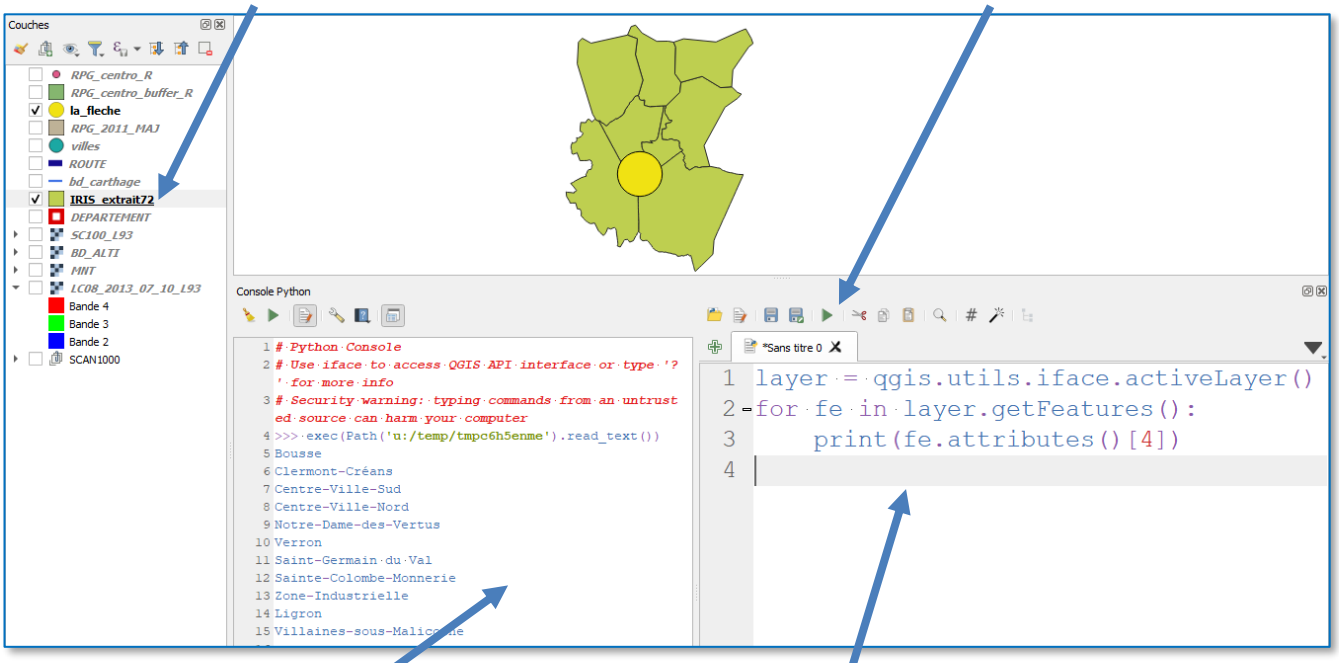


Dans un fichier python, faire une boucle qui imprime le nom de tous les IRIS

Le 5e champ de la table des IRIS, Nom_Iris, est le nom de l'Iris

Couche Iris sélectionnée

Exécuter



Résultat dans la console les noms des Iris le code python

Enregistrer le script

Remarquer le dossier par défaut où sont enregistrés les scripts :

Le plus souvent un dossier dans le profil utilisateur, on peut changer ce dossier dans les réglages de QGIS : Menu > Préférences > Options

Clique sur le bouton « play » pour exécuter le script

6.2.2. Accéder aux couches, objets et attributs en python avec iface

On a déjà vu :

- *Layer = Iface.activeLayer* qui donne la couche active
- *Layer.getFeatures()* un énumérateur qui renvoie la liste des objets d'une couche que l'on peut traiter dans une boucle For.
- *feature.attributes()* renvoie dans 1 tableau les champs attributaires d'un objet

a) Accès aux attributs

Pour accéder à un seul attribut par sa position dans la table ou par son nom :

Feature[2] pour accéder au 3e champ (premier champ = 0)

Feature[«NOM »] pour accéder à ce même champ par son nom

 Faire une boucle qui print le nom des nouvelles régions avec cette syntaxe

Par exemple pour les nouvelles régions :

Champ 0 Champ 1 Champ2

	NOM_REGION	code_supra	NOUVEAUNOM
1	ILE-DE-FRANCE	11	ILE-DE-FRANCE
2	CENTRE	24	CENTRE
3	BRETAGNE	53	BRETAGNE

```
for fe in layer.getFeatures():
    print( fe["NOUVEAUNOM"] )
```

b) Accès aux entités sélectionnées : selectedFeatures()

```
selection = layer.selectedFeatures()
print(len(selection))
for feature in selection:
    # Traitement des entités sélectionnées
```

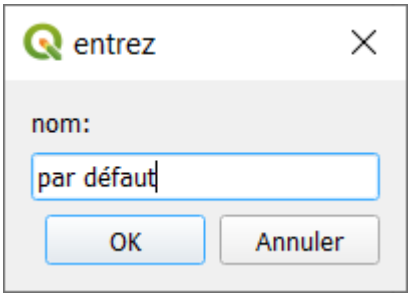
 Faire un script qui calcule et print à la fin, la surface totale de prairies (GRASSLAND) des régions sélectionnées par l'utilisateur : couche REGIONS.SHP

c) Boite de dialogue de type « input text »

Pour demander une valeur à l'utilisateur

```
from qgis.PyQt.QtWidgets import *

qid = QInputDialog()
text, ok = QInputDialog.getText(qid, "entrez ", "nom:", QLineEdit.Normal, "par défaut")

print(text)
print(ok)
```

text contient le texte saisi par l'utilisateur
ok vaut True si l'utilisateur a cliqué sur le bouton OK



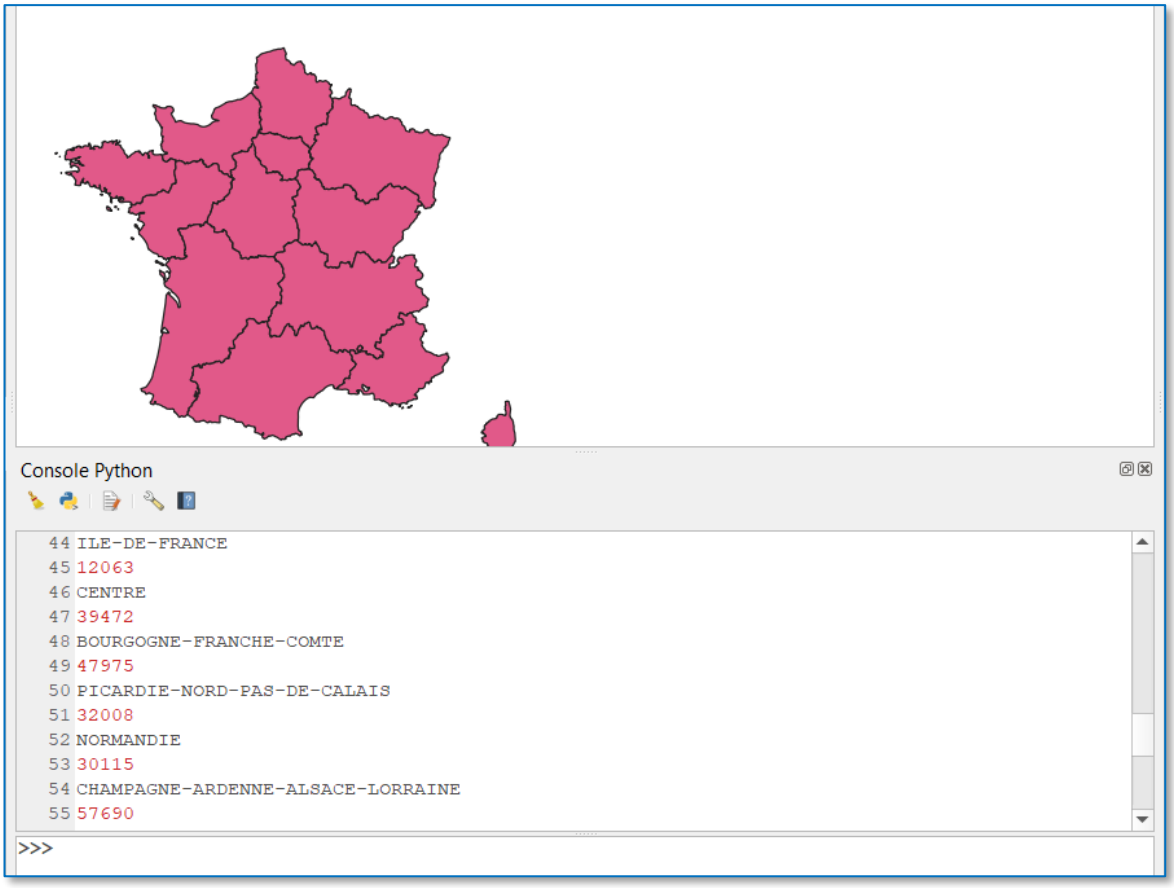
d). Accès à la géométrie

Faire un script qui calcule et print la surface (arrondi au km2) des nouvelles régions

Remarquez que dans la table des nouvelles régions il n'y a pas de champ « surface »

La fonction .geometry() renvoie la geometrie d'un objet (feature)
La fonction area() renvoie la surface en unités de la couche

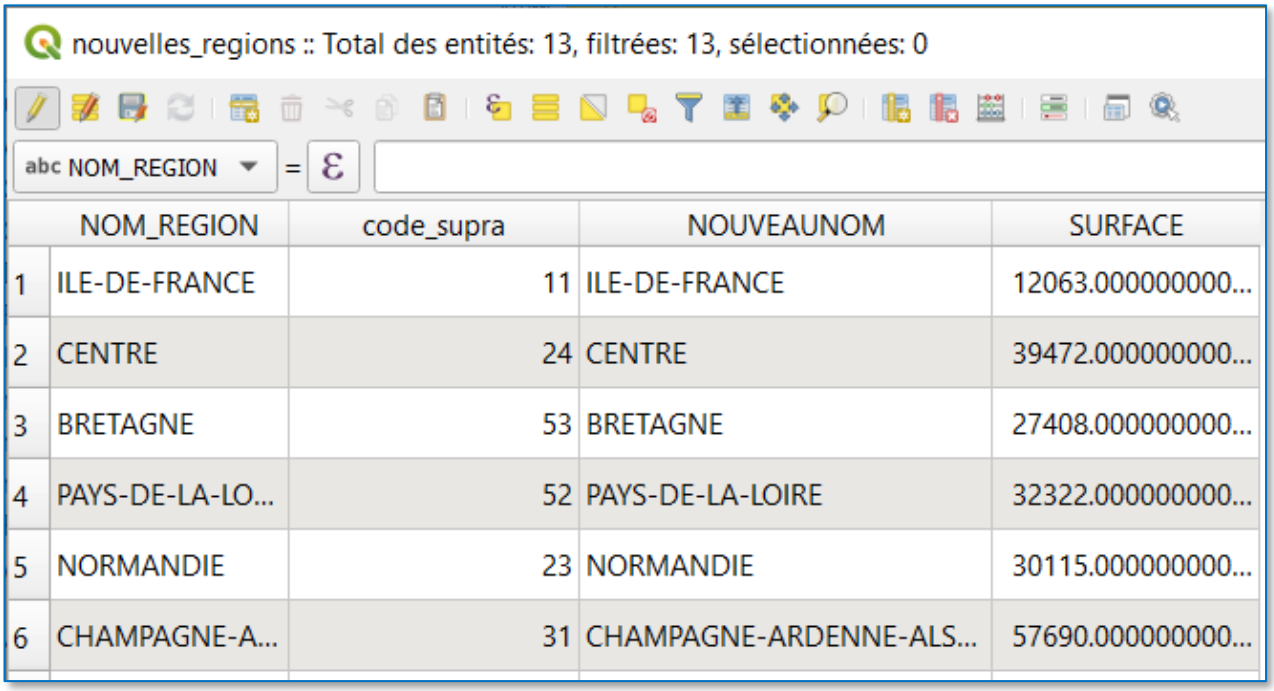
```
layer = iface.activeLayer()
for fe in layer.getFeatures():
    geom = fe.geometry()
    print(fe["NOUVEAUNOM"])
    print(round(geom.area() / 1000000) )
```



e). Calculs dans un champ

Dans les nouvelles régions faire un script qui créé un champ surface et le remplit avec la surface du polygone

```
layer = iface.activeLayer()

layer_provider=layer.dataProvider()
layer_provider.addAttributes([QgsField("SURFACE",QVariant.Double)])
layer.updateFields()

layer.startEditing()

for fe in layer.getFeatures():
    id=fe.id()
    geom = fe.geometry()
    surface = round(geom.area() / 1000000)
    attr_value={3:surface}
    layer_provider.changeAttributeValues({id:attr_value})

layer.commitChanges()
```



nouvelles_regions :: Total des entités: 13, filtrées: 13, sélectionnées: 0

abc NOM_REGION = ε

	NOM_REGION	code_supra	NOUVEAUNOM	SURFACE
1	ILE-DE-FRANCE	11	ILE-DE-FRANCE	12063.000000000...
2	CENTRE	24	CENTRE	39472.000000000...
3	BRETAGNE	53	BRETAGNE	27408.000000000...
4	PAYS-DE-LA-LO...	52	PAYS-DE-LA-LOIRE	32322.000000000...
5	NORMANDIE	23	NORMANDIE	30115.000000000...
6	CHAMPAGNE-A...	31	CHAMPAGNE-ARDENNE-ALS...	57690.000000000...

6.3. Code processing.run() dans la console python (Toolbox processing)

Dans beaucoup cas on voudra seulement lancer une fonction de la toolbox avec des couches en entrée et en sortie, sans accéder aux données objet par objet (interface iface), la syntaxe **processing.run()** va permettre de faire cela. C'est une alternative assez simple au model builder pour faire un script, avec en plus les capacités de Python.

Faire un script qui extrait dans le canton de La Flèche, les parcelles agricoles en Blé tendre.

■ Affichez les 2 couches dont on va se servir :

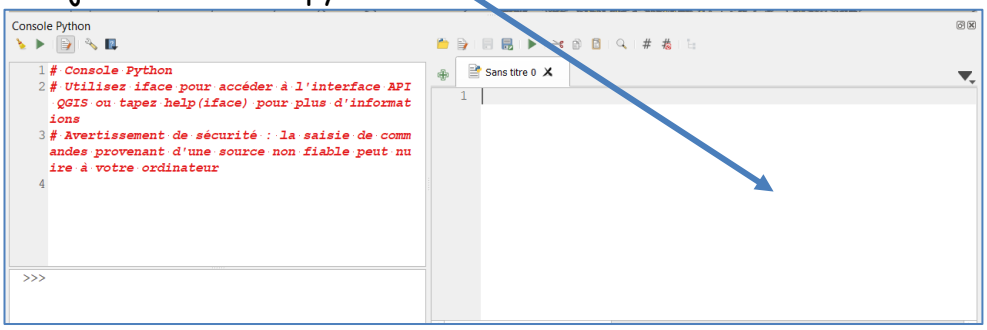
dossier : LA_FLECHE » RPG_2011 » **RPG_2011_MAJ**

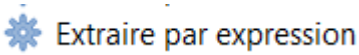
dossier : LA_FLECHE » GEOFLA_FRANCE » CANTON.SHP

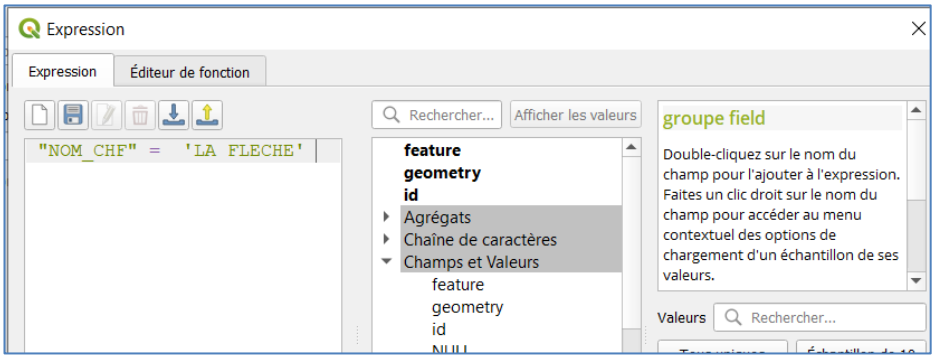
La méthode : on réalise chaque étape dans QGIS « à la main » et on récupère le code python correspondent depuis l'historique des traitements, pour l'ajouter à notre script.

Ouvrir la console python : et Afficher l'éditeur de code avec

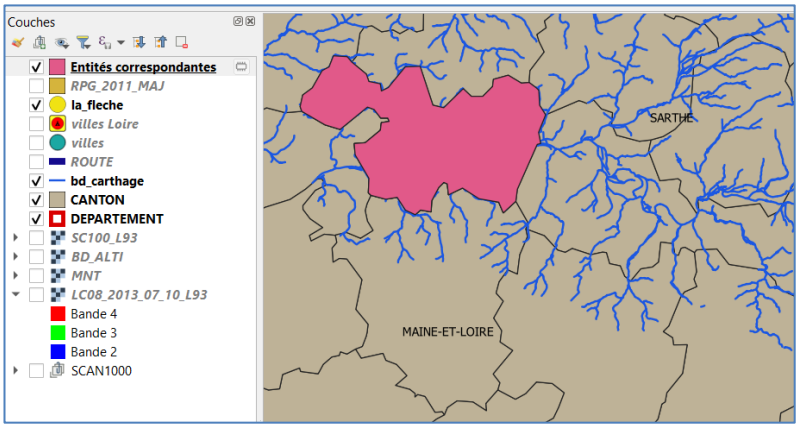
On ajoutera le code python ici



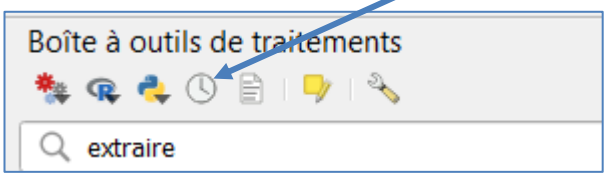
Etape 1 : Extraire le canton de la Flèche 



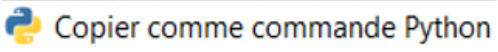
Une nouvelle couche du canton de la flèche a été créé :

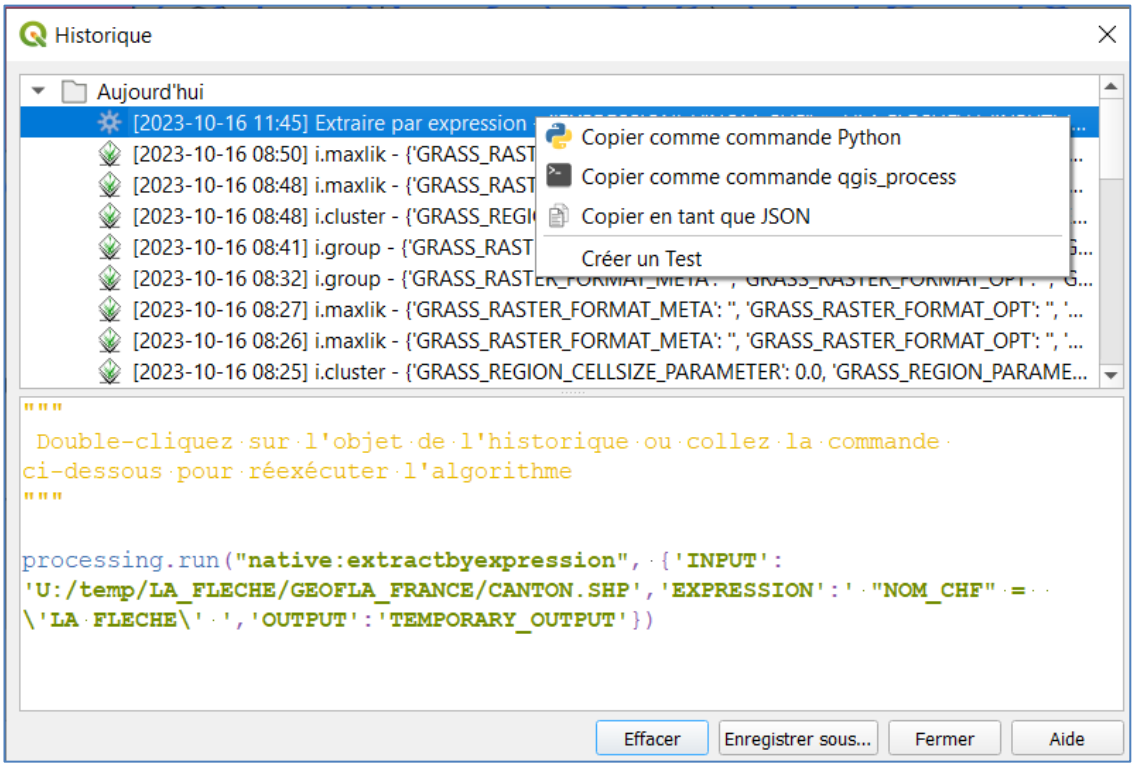


Pour récupérer le code python correspondent à cette opération, ouvrir l'historique de la boite à outils de traitements : 

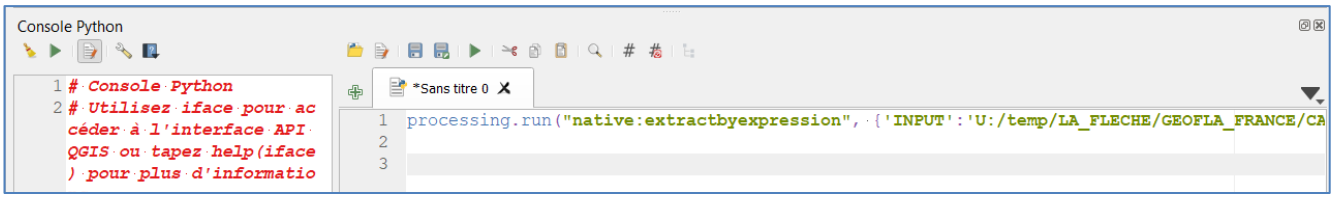


C'est la liste de toutes les dernières opérations effectuées dans la boite à outils, avec leurs paramètres, cliquer bouton droit sur Extraire par expression, puis





Coller dans la fenêtre script de la console python :



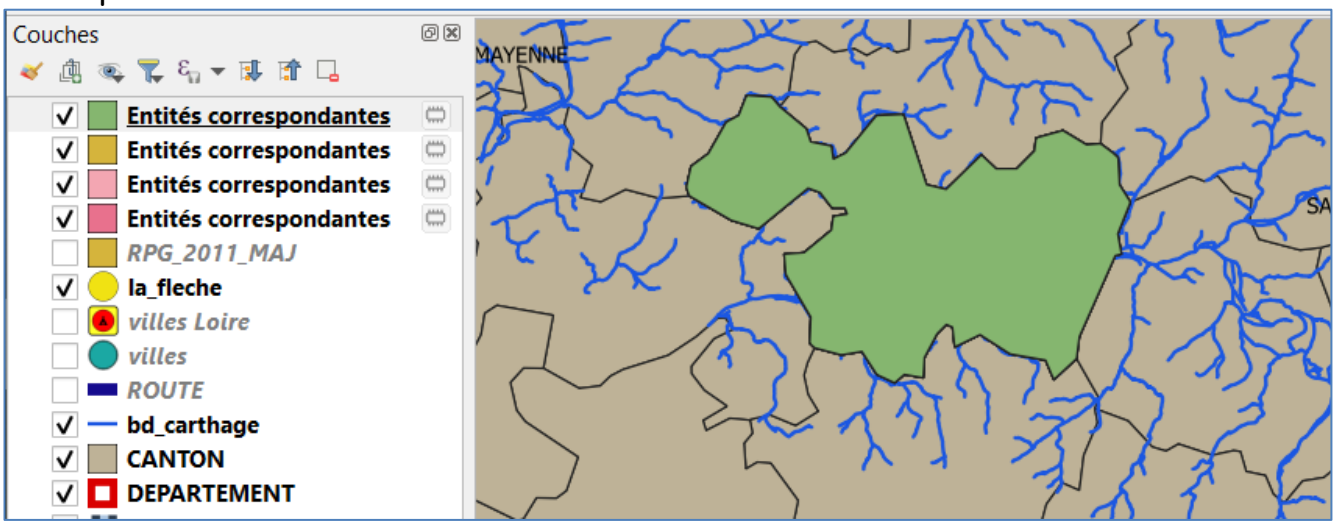
Remarquer bien la syntaxe 'OUTPUT' : 'TEMPORARY_OUTPUT' qui dit que le résultat est mis en couche temporaire.

Changer processing.run en processing.runAndLoadResults
Run ne charge pas le résultat dans le projet contrairement à runAndLoadResults()

C'est juste à cette étape pour voir le résultat, à la fin quand le script sera terminé, on peut mettre runAndLoadResults que pour les résultats finaux, et en run pour les étapes intermédiaires.

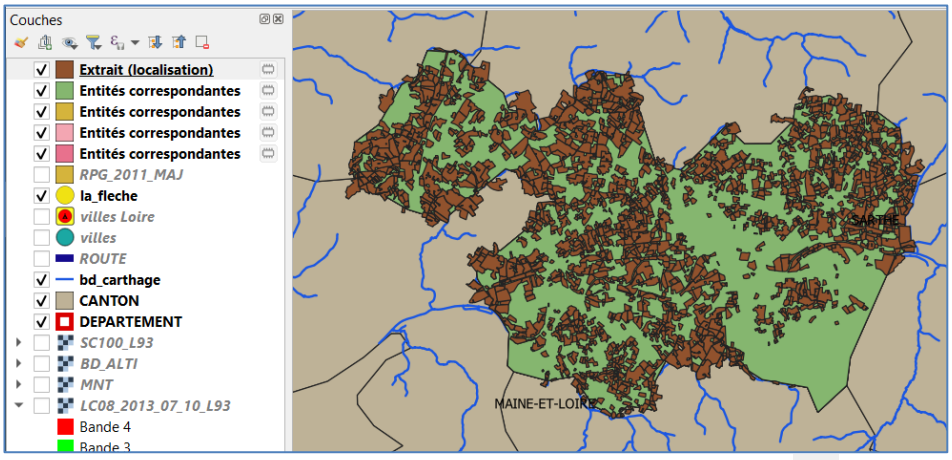
Enregistrer le script

Clique sur le bouton « play » pour exécuter le script :
A chaque fois que l'on clique sur play, une couche temporaire « Entités correspondantes » est créé

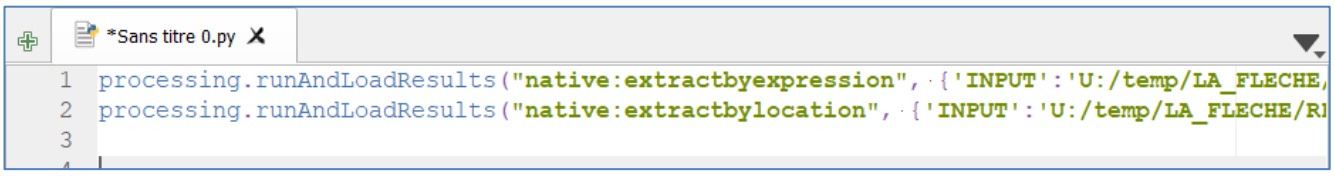


Etape 2 : Extraire les parcelles du canton Extraire par localisation

On prend comme relation géographique, par Exemple les parcelles qui intersecte le canton



On récupère le code de cette étape dans l'historique pour l'ajouter au script



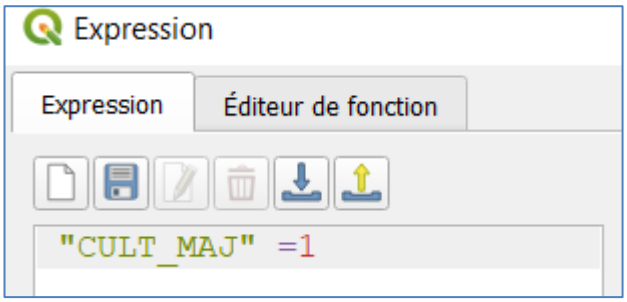
Pour que les deux étapes s'enchainent bien on va indiquer que la sortie de la première étape est une des entrées de la seconde (en couche temporaire):

Modifier la première ligne avec ce code : canton = processing.runAndLoadsResuts(…
La variable canton ainsi créé permet d'accéder aux paramètres du traitement dont par exemple la valeur de la couche de sortie : canton['OUTPUT']

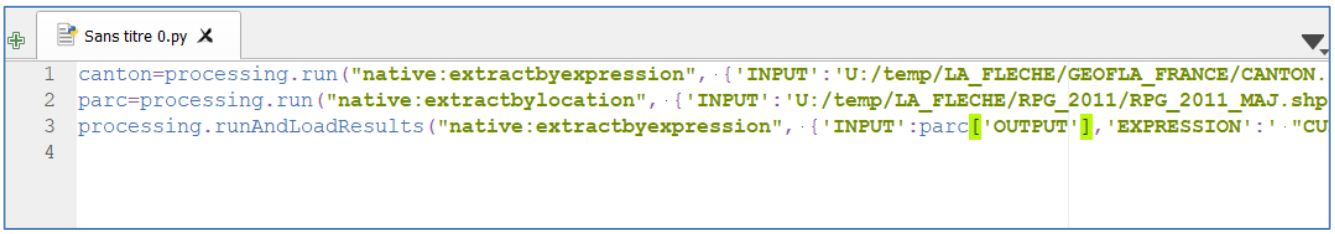
Modifier la seconde ligne pour que le paramètre de couche 'intersect' face référence à la couche canton précédente : 'INTERSECT':canton['OUTPUT']

Remarquer la syntaxe **'nom paramètre' : 'valeur paramètre'**

Etape 3 : Extraire les parcelles de blé tendre (CODE RPG = 1)  Extraire par expression

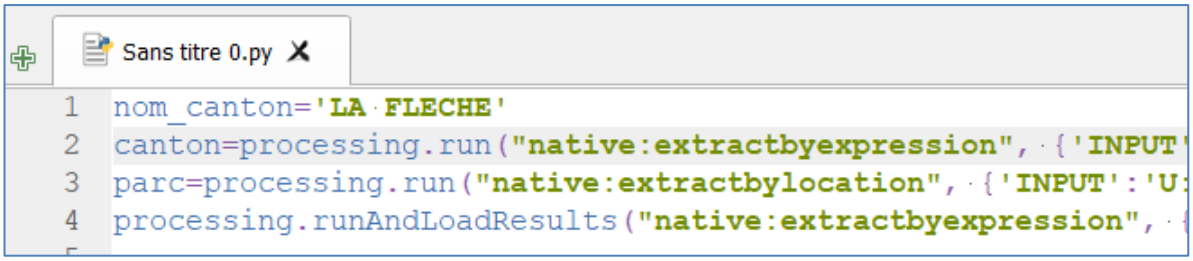


Comme précédemment faire le lien entre sortie de l'étape précédente et entrée de cette dernière étape.
Basculer aussi les étapes intermédiaires en « run » simple et non runAndLoadResults



 Paramétrer le nom du canton dans le script précédent

Pour cela ajouter une variable nom_canton en début de script pour coder le nom du canton :

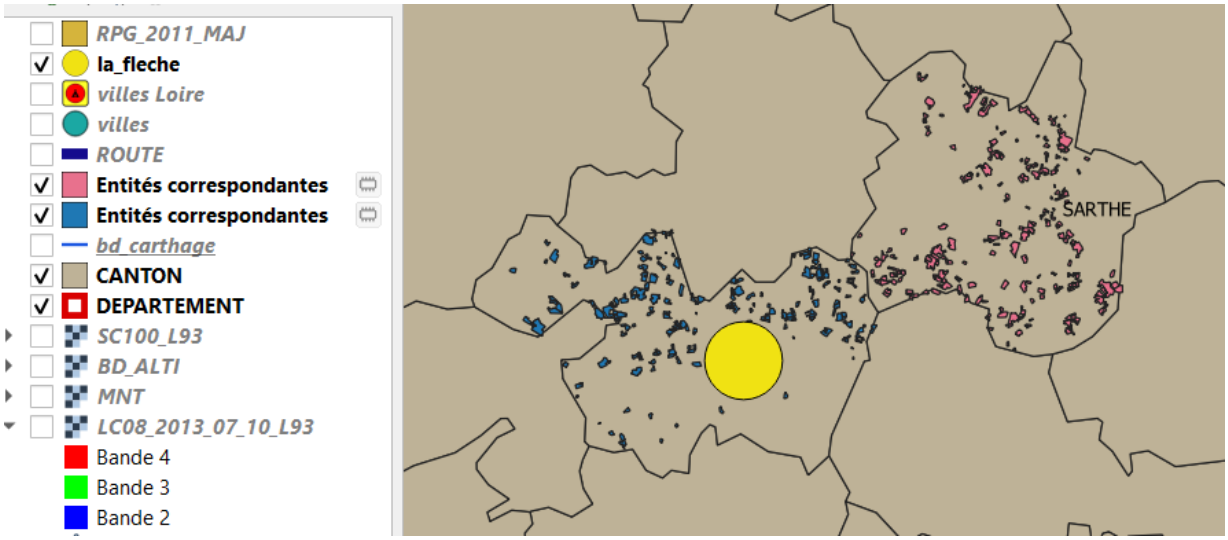


Attention à la syntaxe de l'expression de requête dans la seconde ligne, qui doit combiner du texte constant avec la variable nom_canton :

```
'EXPRESSION':' "NOM_CHF" = \'' + nom_canton + '\' '
```

L'opérateur python « + » pour concaténer des chaines de caractère.

Tester en choisissant pour nom_canton, le canton de PONTVALLAIN



Processing.execAlgorithmDialog() permet d'exécuter une fonction avec sa boite de dialogue et donc que l'utilisateur choisisse les paramètres interactivement.

Transformer votre paramètre précédent en une liste de contons

*Sans titre 0.py

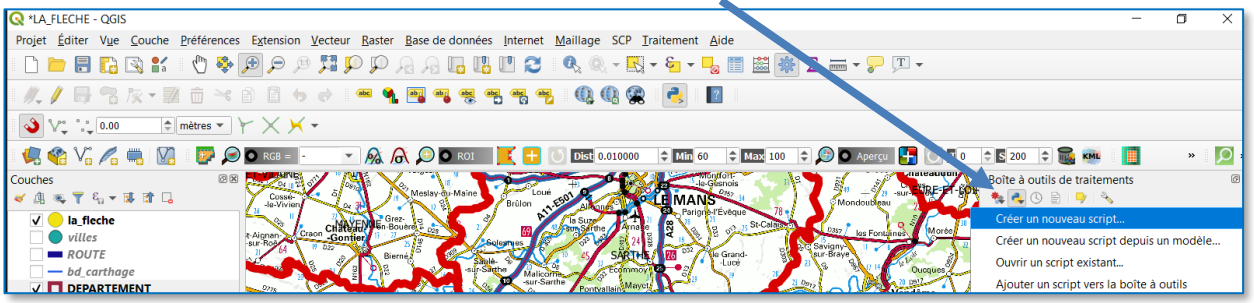
```
1 list_canton=['LA FLECHE','PONTVALLAIN','MALICORNE-SUR-SARTHE']
2 for nom_canton in list_canton:
3     canton=processing.run("native:extractbyexpression", {'INPUT':'U:/tem
4     parc=processing.run("native:extractbylocation", {'INPUT':'U:/temp/LA
5     processing.runAndLoadResults("native:extractbyexpression", {'INPUT':
6
```

Un script consignant les étapes détaillées d'un traitement (ou un modèle de Model Builder) est un outil important de **traçabilité**, c'est-à-dire qui permet de conserver la trace détaillée d'une méthode, ce qui permettra aussi de garantir qu'on appliquera bien la même méthode sur un autre territoire ou à une autre date.

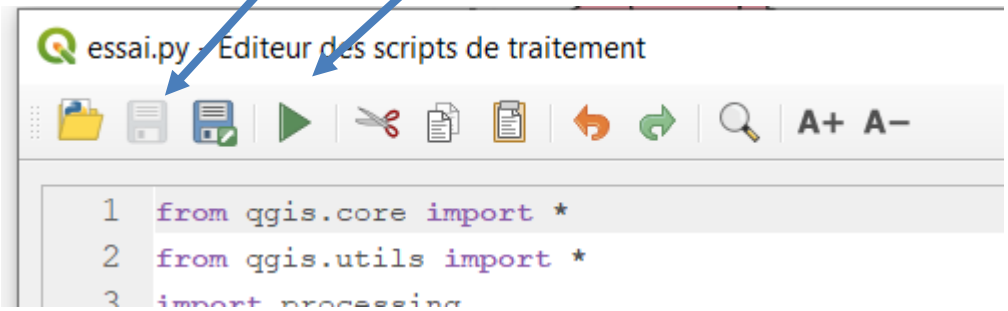
6.4. Code dans un script Python dans la boite à outils Traitement : PyQGIS

6.4.1. Création d'un script comme dans la console

Créez un nouveau script python avec le bouton  dans la boite « Traitements »



Dans la fenêtre Sauver Exécuter 



6.4.2. Les « import » python en début du script

Pour pouvoir utiliser certains fonctions internes de QGIS ils faudra charger en début de script certaines librairies, par exemple les liens minimums sont:

```
from qgis.core import *
from qgis.utils import *
import processing
```

6.4.3. processing.run() : les algorithmes de la boite de Traitement dans un script :

Le plus souvent, pour les scripts les plus simples on ne ferra qu'enchainer des fonctions de la boite à outils de traitement.

 Créer un script qui appelle la fonction centroïde de QGIS sur la couche sélectionnée

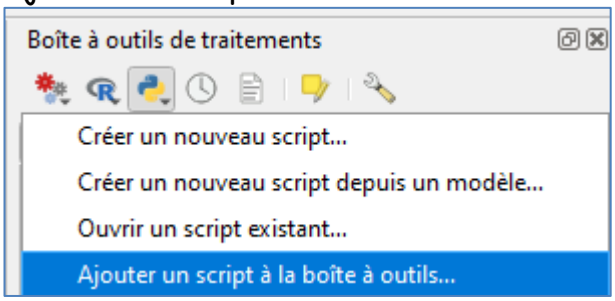
```
from qgis.core import *
from qgis.utils import *
import processing

layer = iface.activeLayer()
fichier = layer.dataProvider().dataSourceUri().split("|")[0].upper()
fichier_out = fichier.split(".SHP")[0]+"_CENTRO.SHP"

processing.runAndLoadResults('native:centroids',{'INPUT':fichier,'ALL_PARTS':False,'OUTPUT':fichier_out})
```

6.4.4. « décorateur » @alg : créer un script dans la boite de traitements :

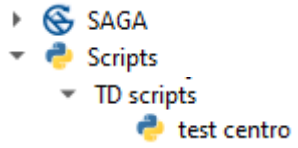
Faire une nouvelle version du script précédent dans la boite à outil de traitement à l'aide de la syntaxe de décorateur @alg

```
from qgis import processing
from qgis.processing import alg


@alg(name='test_centro', label='test centro',group='TDscripts', group_label='TD scripts')
@alg.input(type=alg.SOURCE, name='INPUT', label='Input  layer')
@alg.input(type=alg.VECTOR_LAYER_DEST, name='OUTPUT',label='vecteur output')


def test_centro(instance, parameters, context, feedback, inputs):

    """

    Description of the algorithm.

    (If there is no comment here, you will get an error)

    """

    res=processing.run('qgis:centroids',
{'INPUT':parameters['INPUT'],'ALL_PARTS':False,'OUTPUT':parameters['OUTPUT']},is_child_algorithm=True,
context=context,feedback=feedback)



    return {'OUTPUT': res['OUTPUT']}
```

Ajouter ce script dans la boite à outils de Traitements avec le bouton  :



Dans la boite à outils on voit sous « Python » notre script dans sa catégorie



Un script intégrer dans la boite à outils aura une boite de dialogue associée

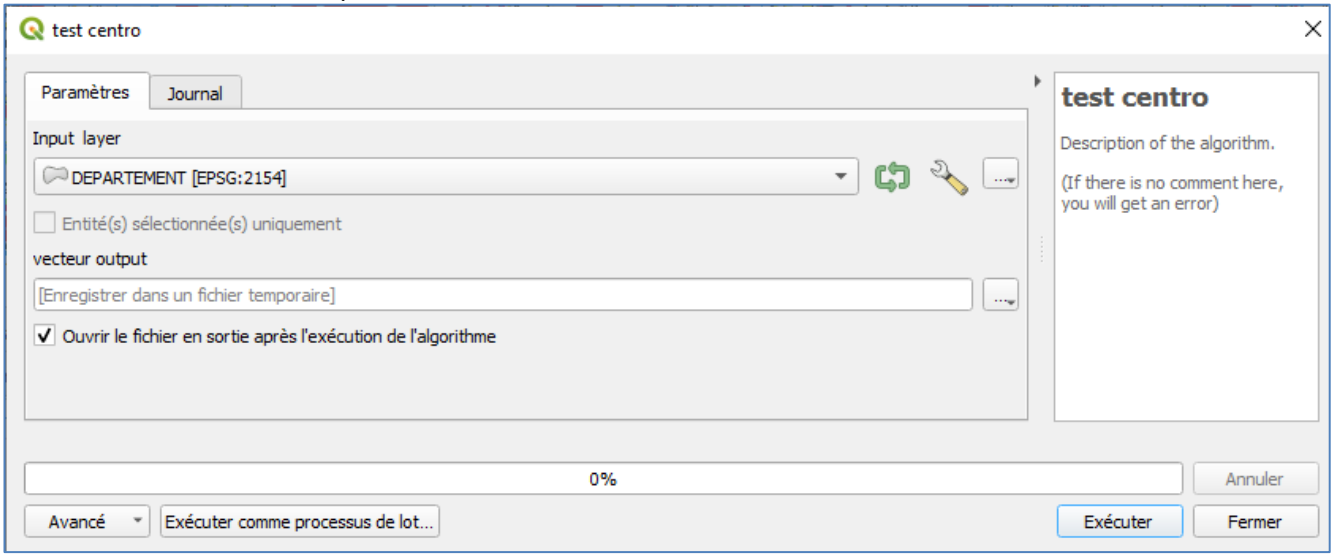
import alg pour importer la syntaxe @alg

@alg(name='test_centro', label='test centro',group='TDscripts', group_label='TD scripts')
Défini le nom du script et le nom du groupe où il est placé dans la toolbox

@alg.input pour les paramètres d'entrée / sortie

Cliquer sur le script pour le lancer :

Une boite de dialogue est automatiquement créée, comme pour toutes les fonctions internes de QGIS. On peut choisir les couches en entrée et en sortie.



⚠ Attention des erreurs de syntaxe dans le script peuvent bloquer son importation dans la toolbox, voir même planter QGIS, par exemple le commentaire dans la définition de la fonction (def) est obligatoire.

Les scripts importés dans la ToolBox sont copiés dans ce dossier, dans votre profil :
jmgilliot › AppData › Roaming › QGIS › QGIS3 › profiles › default › processing › scripts
C'est ce fichier qui est lié à la toolbox maintenant.

Afficher des messages dans la boite de dialogue du script

L'objet « feedback » possède des méthodes d'affichage de messages telle que feedback.pushInfo

Ajouter des messages dans le script :

```
23
24      feedback.pushInfo("***** information ******")
25
26      return {'OUTPUT': res['OUTPUT']}
27
28
```

Regarder dans la boite de dialogue, après exécution du script :

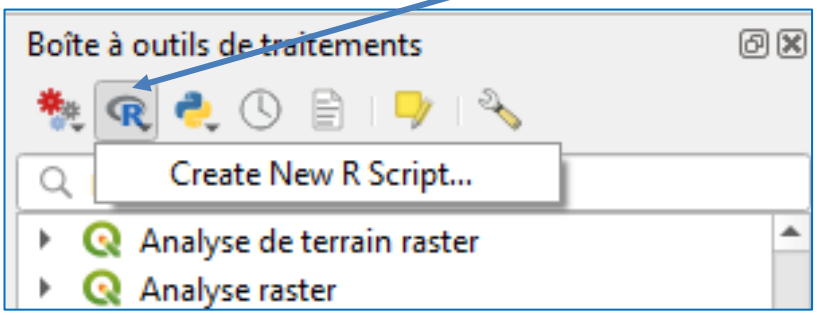
Dans le volet « Journal »

6.4.5. Script R dans la Toolbox de traitement

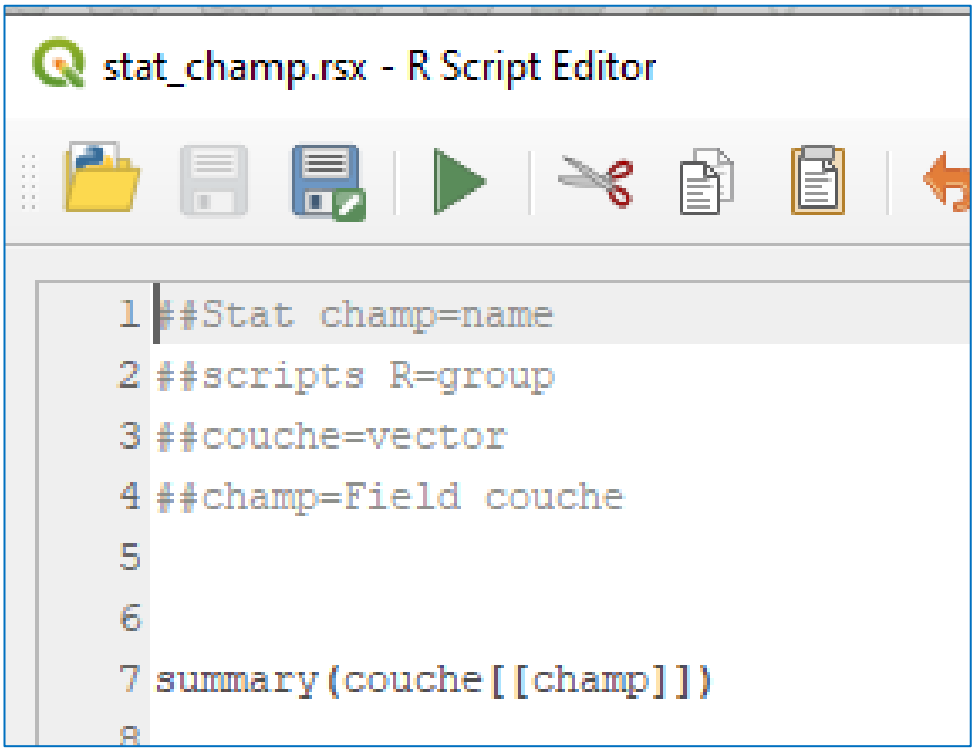
On peut aussi intégrer des scripts en langage R dans la Toolbox de traitement.

Vérifier que l'extension Processing R Provider est bien installée.

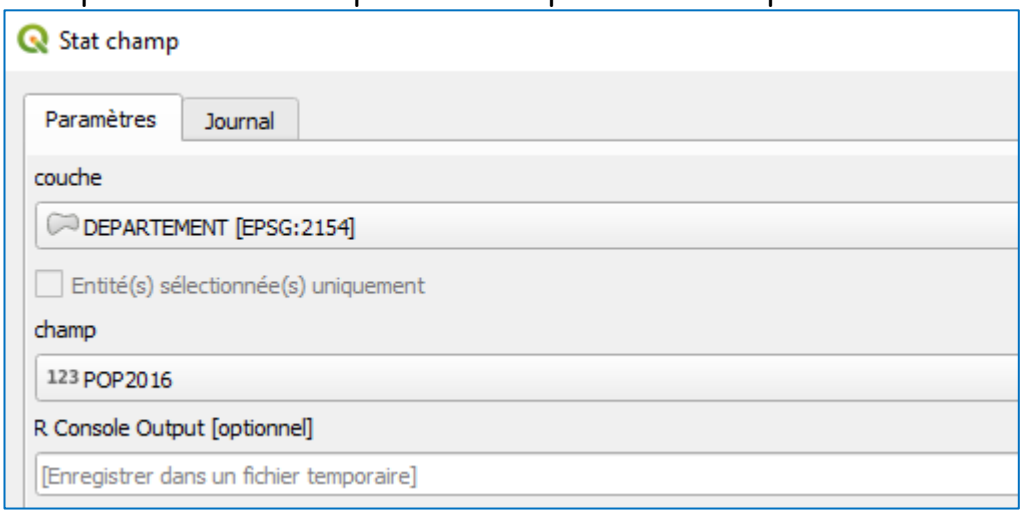
On peut ajouter un script R en cliquant ici dans la Toolbox Traitement



Faire un script R qui affiche les statistiques de base d'un champ d'une couche.

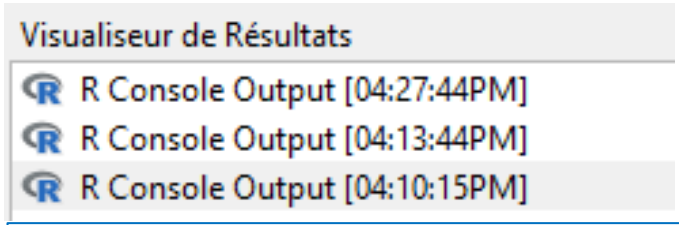


est le symbole de « décoration » pour entrer les paramètres du script.
Nom de variable=propriété
Stat champ=name est le nom du script
Group est le groupe de scripts
Couche= vecteur la variable couche reçoit une couche de type vecteur
Champ est une variable pour un champ de la couche précédente



Les résultats dans le Journal

Ou en cliquant sur le lien dans le volet



R Output

```
[1] "C:/Users/jmgilliot/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/rlibs/sf"
Linking to GEOS 3.9.1, GDAL 3.3.2, PROJ 7.2.1; sf_use_s2() is TRUE
[1] "C:/Users/jmgilliot/AppData/Roaming/QGIS/QGIS3/profiles/default/processing/rlibs/raster"
Le chargement a nÃ©cessitÃ© le package : sp
Min. 1st Qu. Median Mean 3rd Qu. Max.
75784 301306 537809 672965 847263 2617319
```

À retenir :

Partie 6. Script python et R dans l'interface de QGIS

- On peut saisir et exécuter du code python à différents endroits dans l'interface QGIS :
  - Directement dans la console python
  - Dans l'éditeur de code de la console python
  - Intégrer son code comme un nouvel outil dans la boite de traitement

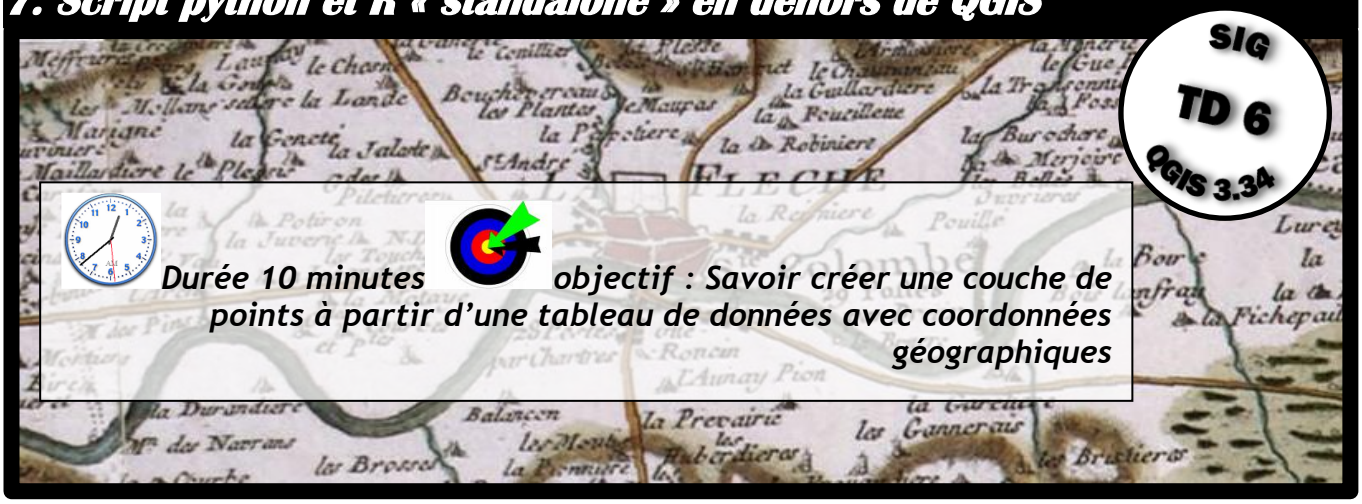
- La façon le plus simple d'utiliser les opérations internes de QGIS est par la fonction processing.run(), mais ne permet pas de manipuler le projet et les objets géographiques (polygones par ex) individuellements.

- L'interface iface permet quant à elle cette manipulation du projet et des objets individuellement

- La syntaxe décorateur @alg permet de faire des scripts intégrés dans la boite à outils avec une boite de dialogue.

- L'extension Processing R provider permet d'ajouter des scripts R dans la boite à outils, on peut ainsi accéder en langage R aux objets du projets et utiliser tous les traitements R.

7. Script python et R « standalone » en dehors de QGIS



7.1. qgis_process : Code « processing » dans un script externe à QGIS

Comme dans le cas précédent pour faire des scripts n'utilisant que des appels à des fonctions de la boite à outils de traitement, mais cette fois dans un script externe à QGIS, sans que QGIS soit ouvert.

Depuis quelques versions de QGIS a été introduit **qgis_process** un exécutable dans l'installation de QGIS, qui permet directement depuis la ligne de commande ou depuis un fichier script, de lancer des fonctions de la boite à outils de traitements, sans avoir à lancer QGIS lui-même. Utilisable potentiellement depuis n'importe quel langage qui accepte des appels systèmes (pour appeler qgis_process) : batch windows (.cmd .bat), script shell Linux (.sh), script python, script R, visual basic dans Excel etc …

Consulter les explications détaillées sur le site de QGIS :
https://docs.qgis.org/3.34/fr/docs/user_manual/processing/standalone.html

Sous Windows qgis_process.exe est localisé dans le sous dossier QGIS :
apps/qgis-ltr/bin ou apps/qgis/bin

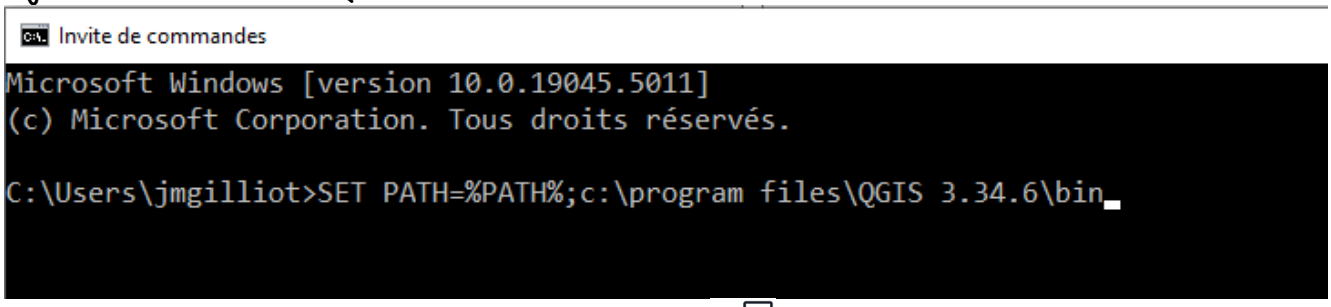
Sous windows on n'appelle généralement pas qgis_process.exe directement, mais un fichier de commande .bat qui initialise aussi les variables QGIS, depuis le sous-dossier QGIS :
Bin/qgis_process-qgis-ltr.bat ou bin/ qgis_process-qgis.bat

Ouvrir une fenêtre de commande (ou powershell) windows

Taper : cmd +

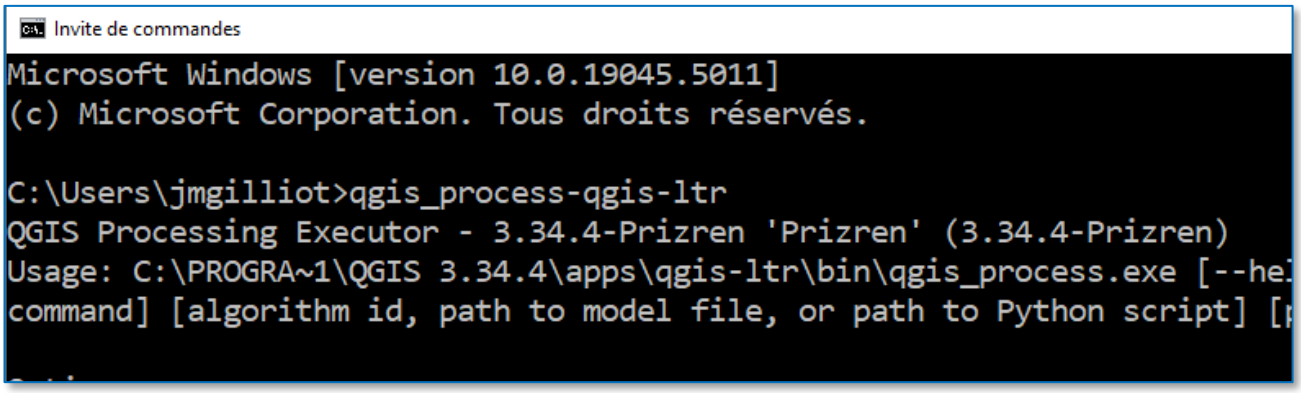
Cela ouvre une fenêtre de commande

Ajouter le chemin de QGIS à la variable windows PATH



Taper dans la fenêtre : qgis_process-qgis-ltr +

ou qgis_process-qgis si la version n'est pas LTR



Les commandes QGIS sont maintenant accessibles en ligne de commandes.

Sous Linux qgis_process est normalement directement accessible, après installation.

7.1.1 qgis_process en ligne de commande

Créer une nouvelle couche des centroïdes des parcelles RPG avec une ligne de commande utilisant qgis_process

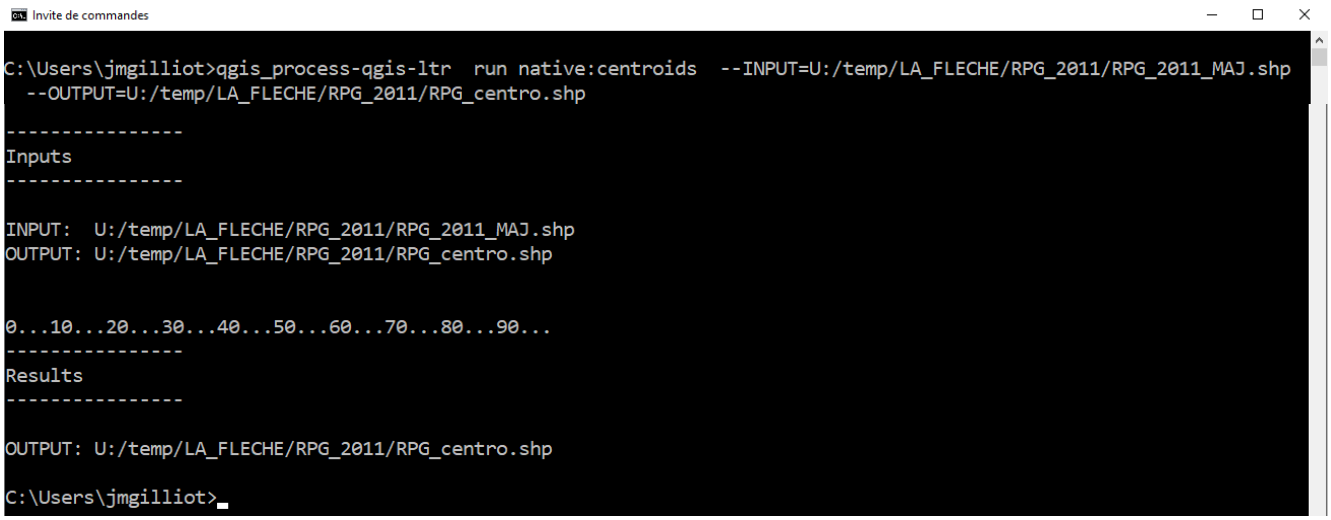
LA_FLECHE → RPG_2011 → RPG_2011_MAJ

Repérer le chemin complet sur le disque des données :
par exemple : U:/temp/LA_FLECHE/RPG_2011

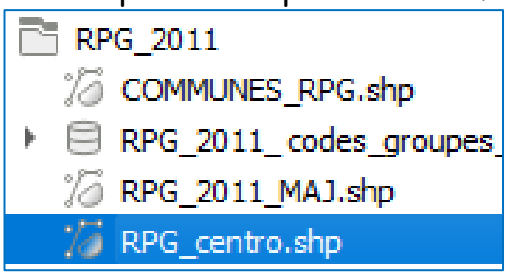
taper cette commande dans la fenêtre de commande (shell) + ↵

qgis_process-qgis-ltr run native:centroids --INPUT=U:/temp/LA_FLECHE/RPG_2011/RPG_2011_MAJ.shp --OUTPUT=U:/temp/LA_FLECHE/RPG_2011/RPG_centro.shp

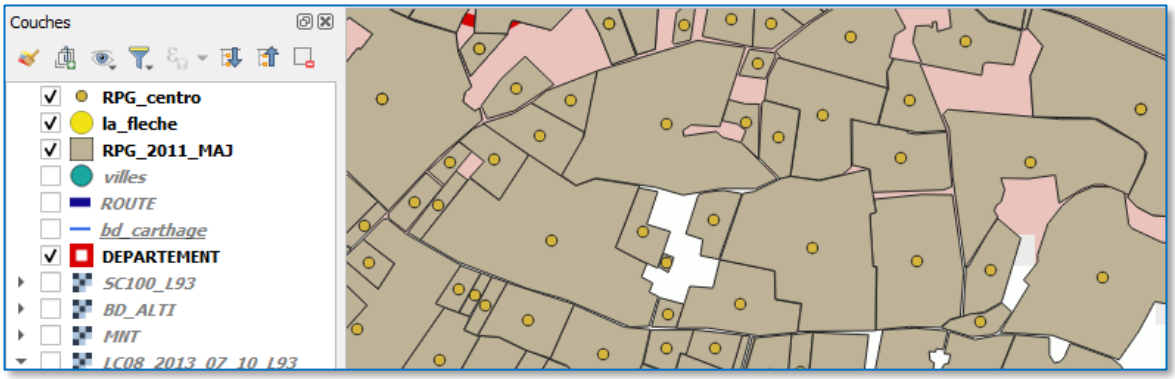
(run pour exécuter une commande)



Dans le panneau explorateur de fichier sous QGIS le fichier résultat est apparu :



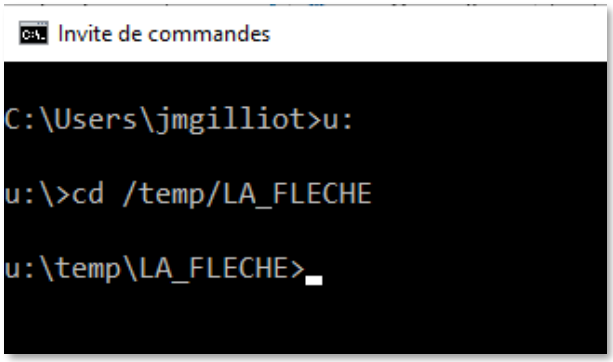
Ajouter le dans le projet QGIS



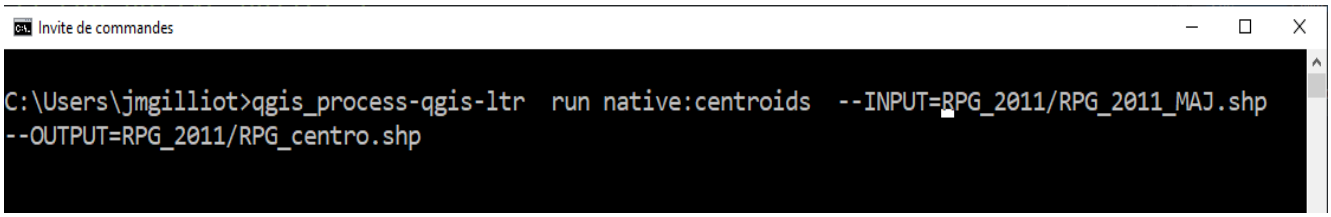
Renouveler le traitement mais en se plaçant dans le dossier projet pour avoir des chemins plus courts vers les données

Dans la fenêtre de commande utiliser l'instruction CD (change directory) pour se mettre dans le dossier de LA_FLECHE

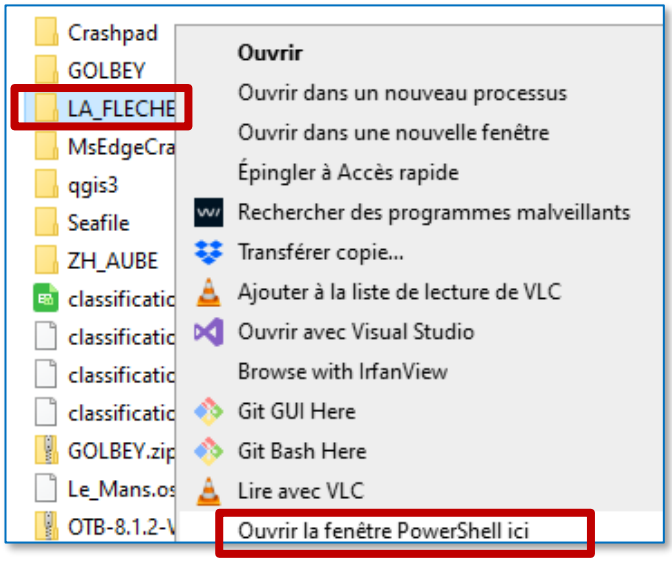
U : pour changer de disque si nécessaire
Puis cd /temp/LA_FLECHE



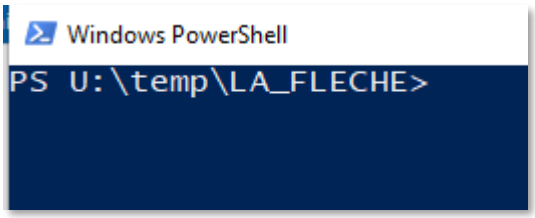
Relancer la commande « qgis_process » précédente avec des chemins relatifs aux données, cela fait des chemins moins long à entrer.



On peut aussi ouvrir une fenêtre de commande à partir de l'explorateur de fichiers Windows, directement dans le dossier LA_FLECHE



Touche Shift + clic droit souris sur le dossier LA_FLECHE



On est dans le bon dossier directement

Choisir Ouvrir la fenêtre ici

7.1.2 qgis_process dans un script python « standalone »

Renouveler le traitement précédent mais depuis un script python

Créer à la racine de LA_FLECHE un fichier texte « script.py » et l'éditer avec un IDE python comme ***spyder (pour python 3)*** par exemple

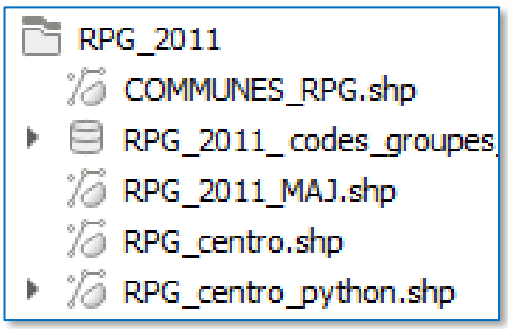
```
1  import subprocess
2
3  # variable avec le chemin vers qgis_process
4
5  qgis_process="c:/program files/QGIS 3.34.6/bin/qgis_process-qgis-ltr.bat"
6
7  # calcul des centroïdes
8  cmd=qgis_process+" run native:centroids \
9      --INPUT=RPG_2011/RPG_2011_MAJ.shp \
10     --OUTPUT=RPG_2011/RPG_centro_python.shp"
11
12 subprocess.run(cmd)
13
```

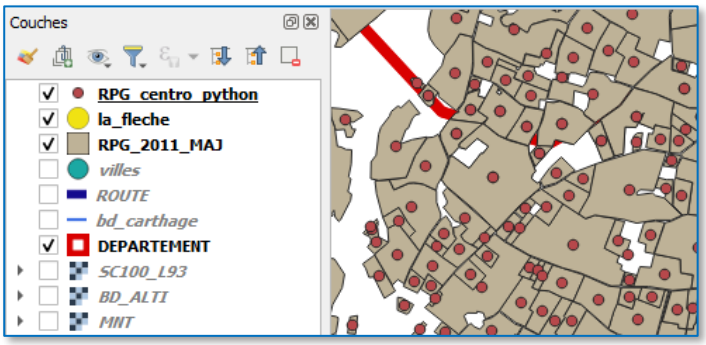
Subprocess est une librairie on y trouve la fonction run qui permet d'exécuter une commande externe à Python, il existe plusieurs fonctions pouvant le faire en python

Cmd est une variable chaine de caractère dans laquelle on prépare la commande à exécuter, remarquer l'antislash « \ » en fin de ligne qui permet d'introduire un saut de ligne pour une meilleur lisibilité du code.

Exécuter le code avec  le résultat est nommé « RPG_centro_python.shp »

Ce résultat apparait dans le dossier RPG_2011:

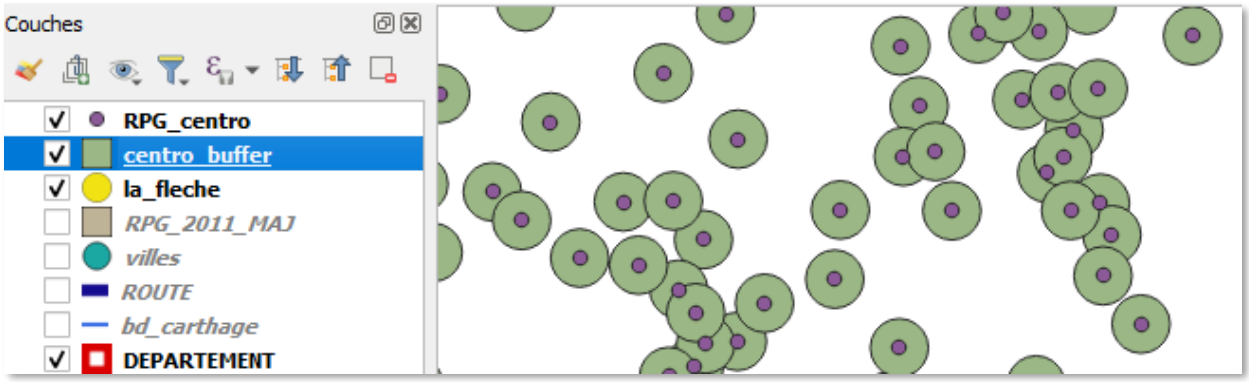




Calculer un buffer de 100m à partir des centroïdes précédent, en passant par un fichier temporaire de centroïde et en enchainant les deux opérations dans le même script python.

```python
import subprocess
import tempfile

# variable avec le chemin vers qgis_process

qgis_process="c:/program files/QGIS 3.34.6/bin/qgis_process-qgis-ltr.bat"

# Création d'un répertoire temporaire

temp_dir = tempfile.TemporaryDirectory()
tdir = temp_dir.name+"/"


# calcul des centroïdes
cmd=qgis_process+" run native:centroids \
    --INPUT=RPG_2011/RPG_2011_MAJ.shp \
    --OUTPUT="+tdir+"RPG_centro_python.shp"
subprocess.run(cmd)

# calcul des buffers
cmd=qgis_process+" run native:buffer \
    --INPUT="+tdir+"RPG_centro_python.shp \
    --DISTANCE=100 \
    --OUTPUT=RPG_2011/RPG_centro_python_buffer.shp"
subprocess.run(cmd)

# Efface le répertoire temporaire
temp_dir.cleanup()
```

La librairie tempfile permet de créer un dossier (ou fichier) temporaire pour le fichier intermédiaire des centroïdes.




On peut récupérer dans une variable des informations, sur la réussite de la fonction et autres messages d'erreur avec une syntaxe du type :
res = subprocess.run()

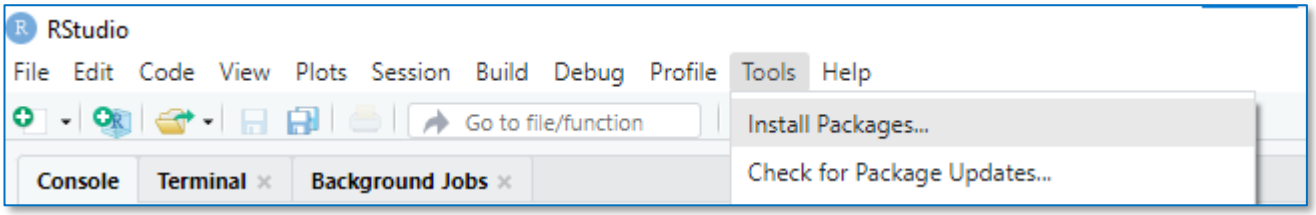
7.1.3 qgis_process dans un script R « standalone »

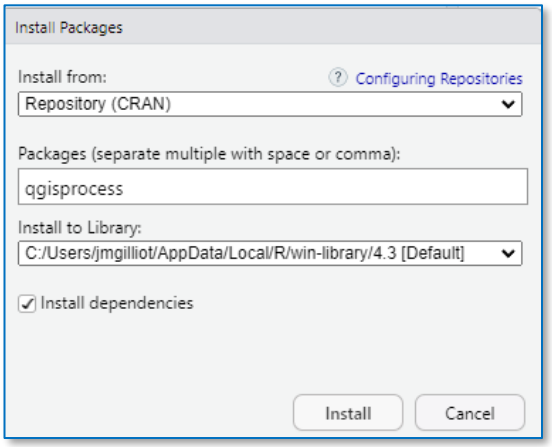
Créer un fichier script.R à la racine du projet et Ouvrir RStudio

Créer une nouvelle couche des centroïdes des parcelles RPG avec une ligne de commande utilisant qgis_process sous R

On pourrait aussi sous R, faire un appel system à qgis_process comme dans les exemples python, mais on va ici utiliser **le package R « qgisprocess »** qui facilite le travail.

Installer le package avec Install Packages



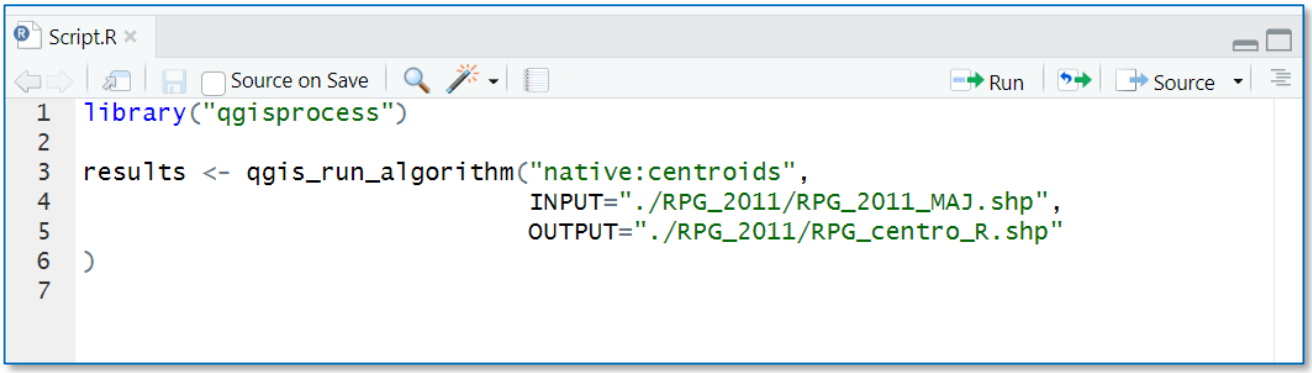


Ouvrir le fichier Script.R

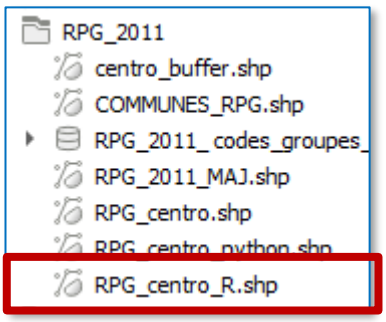
Fixer le répertoire courant de RStudio au dossier du script :

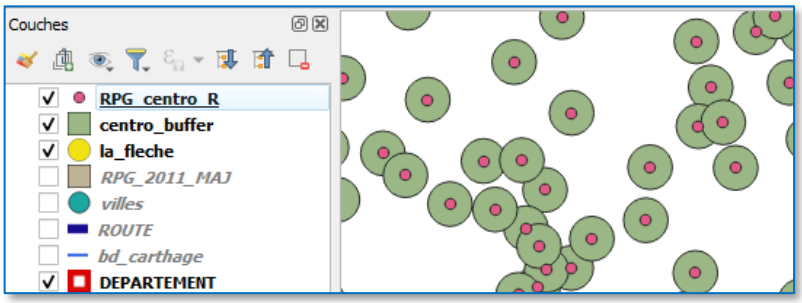
Menu : Session >> Set Working Directory >> to Source File Location

Saisir le code et cliquer sur Source pour exécuter tout le script

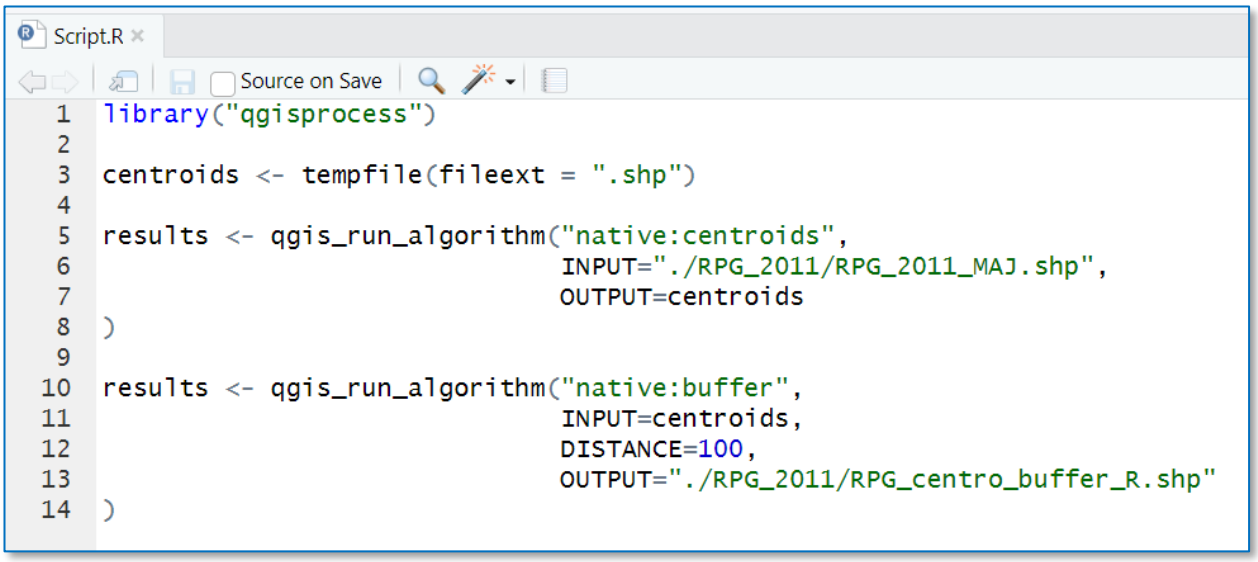


Le fichier RPG_centro_R a bien été créé

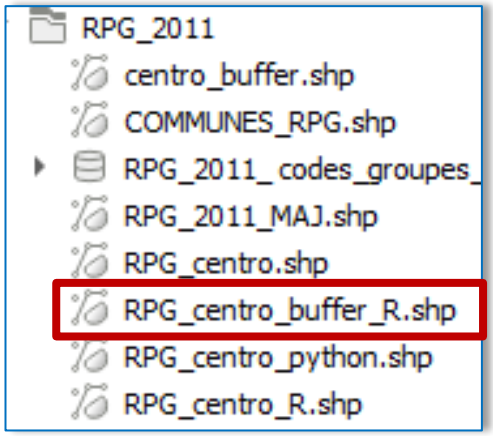


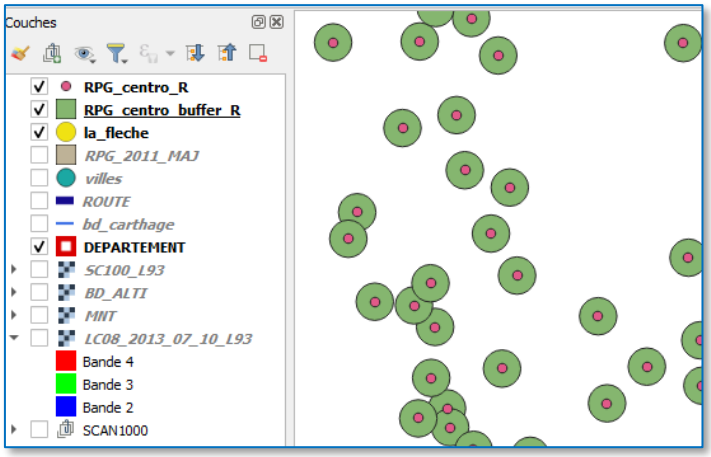


Calculer un buffer de 100m à partir des centroïdes précédent, en passant par un fichier temporaire de centroïde et en enchainant les deux opérations dans le script R.



Remarquer la fonction tempfile() qui permet de créer un nom de fichier temporaire.
Le fichier RPG_centro_buffer_R.shp a bien été créé






Il y a de nombreux packages R pour le traitement des données géographiques, que l'on peut utiliser en complément de qgisprocess, on peut citer :

- Package « sf » (simple feature) pour les données vectorielles
- Package « terra » (anciennement raster) pour le traitement des rasters

On peut trouver dans ces packages des fonctions de traitements géographiques équivalentes à celles de QGIS, par exemple dans le package « sf » une fonction de zone buffer : `Buffer <- st_buffer(points, dist=200)`

7.2 programmes python externe à QGIS

Il est possible de programmer une application externe en python, utilisant toutes les fonctions de QGIS, un peu plus compliquer à faire.

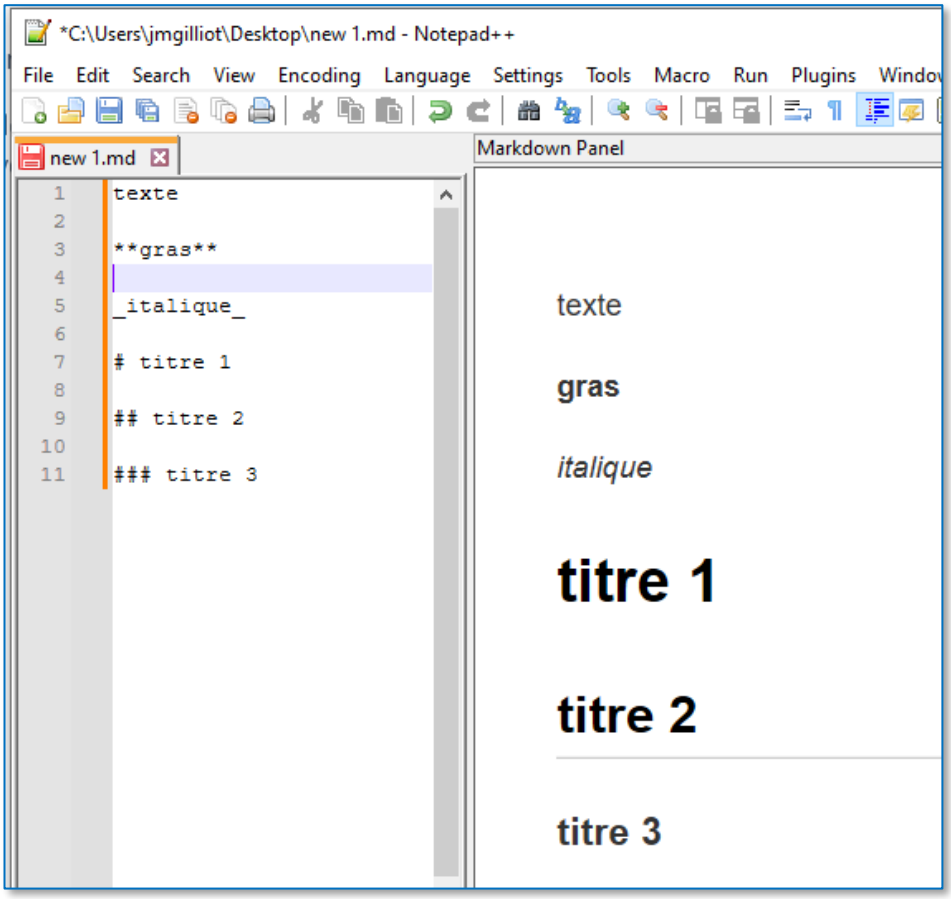
8. Script python et R « standalone » en RMarkdown

8.1 Notion de fichier Markdown

Markdown est un langage permettant d'introduire des balises de mise en forme très simple, en format texte, à l'aide de symboles prédéfinis, un peu à l'image des balises HTML.

Par exemple dans NotePad++ avec le plugin markdown panel

Code Rendu



8.2 Fichier RMarkdown

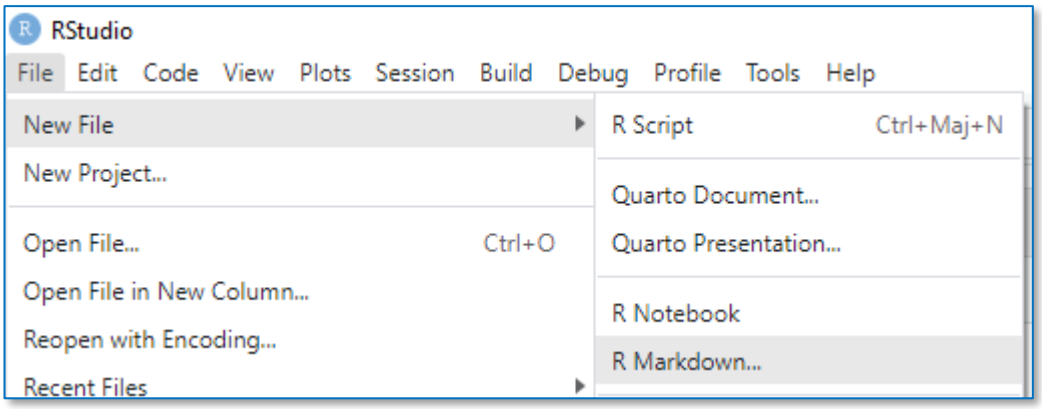
RMarkdown est un format de fichier basé sur Markdown, mais permet beaucoup plus de chose, il a été créé pour R, il permet de générer des **rapports dynamiques**, mélangeant du texte mis en forme, du code informatique, et des graphiques produits par ce code.

L'extension de fichier pour RMarkdown est « .Rmd », le rapport qui est généré dynamiquement est généralement en format : html, Pdf ou Word.

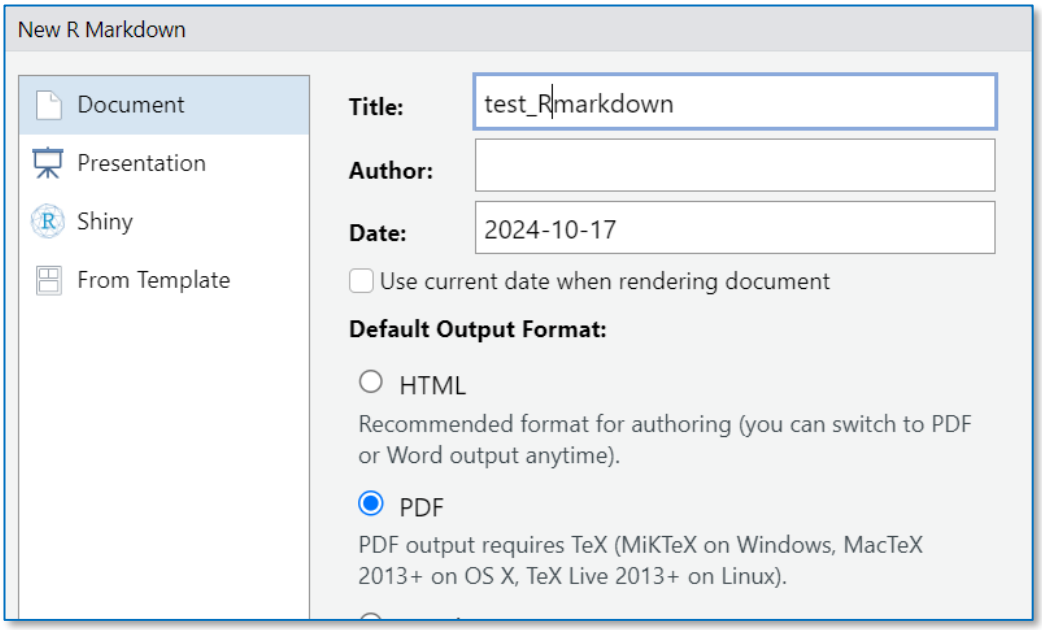
Oucrir RStudio

Vérifier que le package rmarkdown et tinyTex (LaTex) est bien installé

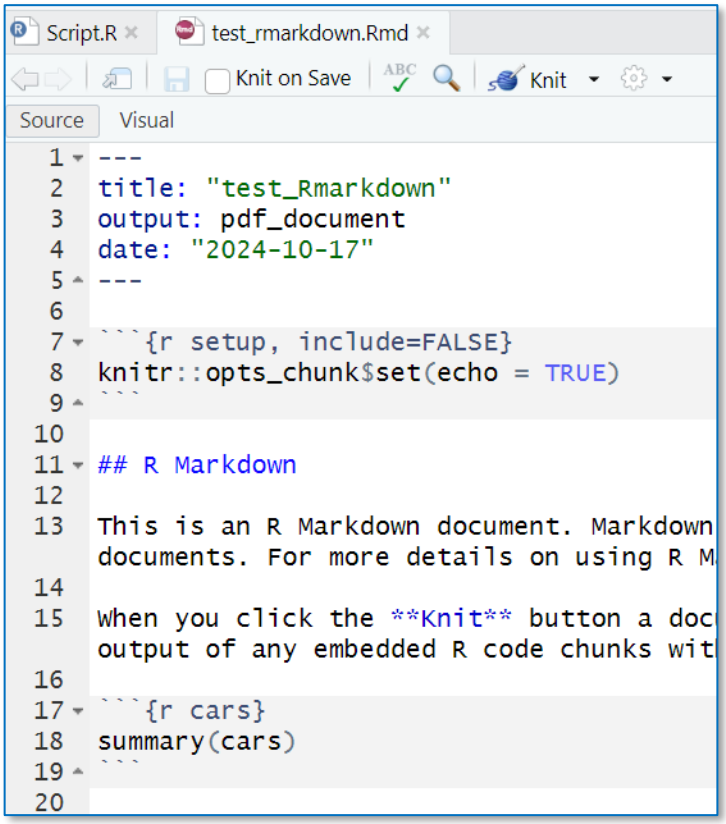
Créer un fichier R Markdown dans RStudio :



On choisit un type « Document » format de sortie en PDF par exemple



Enregistrer dans un fichier .rmd avec



Entete

Chunk R pour setup général

Texte mise en forme = titre

Texte simple

Chunk R

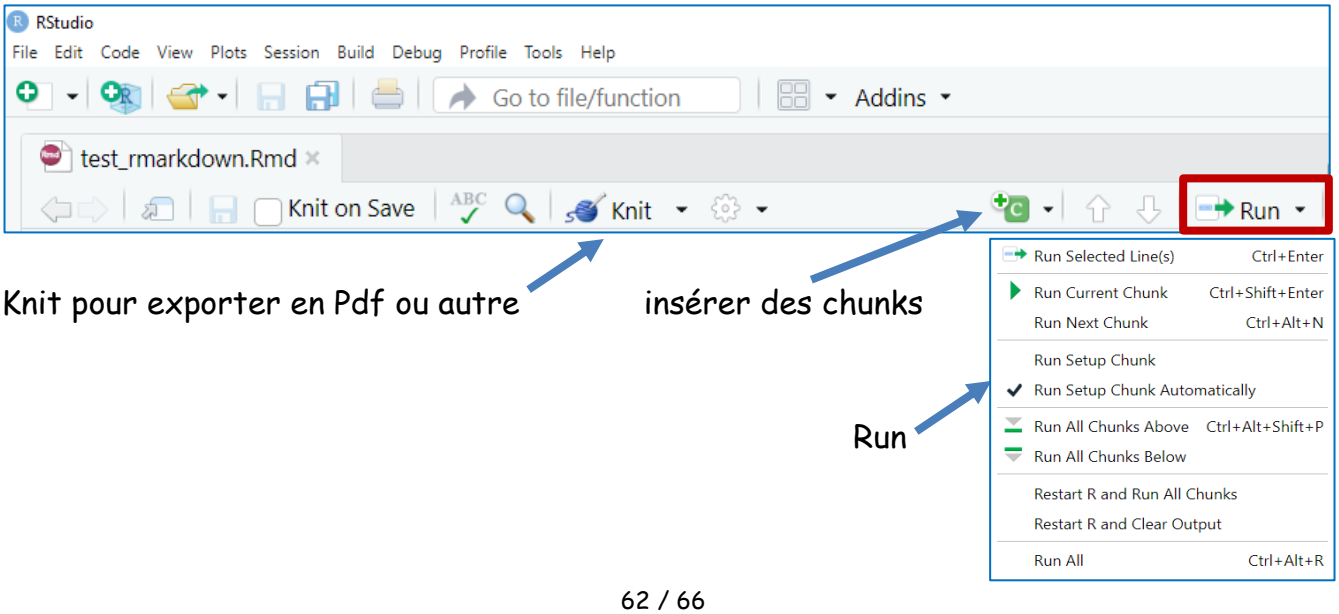
Le fichier mélange du texte, éventuellement mis en forme et des parties nommées « chunk » qui contiennent du code ici en R.

Faire un fichier RMarkdown, qui compte le nombre de parcelles RPG dans les ilots IRIS, en utilisant qgisprocess et qui fait en graphique R pour présenter les résultats.

Utiliser :

« native :centroids » pour convertir les parcelles en points

« native:countpointsinpolygon » pour compter le nombre de points par polygone



Knit pour exporter en Pdf ou autre

insérer des chunks

Run

````
---
title: "test_Rmarkdown"
output: word_document
date: "2024-10-17"
---

```{r setup, include=FALSE,echo=FALSE}

library("qgisprocess")
library("sf")
library("ggplot2")

```

## Traitements QGIS pour le comptage de parcelles

Dans cette partie traitement géographique avec QGIS comptage des *parcelles RPG* par zone IRIS

```{r comptage, warning=FALSE, message=FALSE}



centroids <- tempfile(fileext = ".shp")

results <- qgis_run_algorithm("native:centroids",
                              INPUT="RPG_2011/RPG_2011_MAJ.shp",
                              OUTPUT=centroids
)

results <- qgis_run_algorithm("native:countpointsinpolygon",
                              POINTS=centroids,
                              POLYGONS="Contours_Iris/carto/IRIS_extrait72.shp",
                              OUTPUT="RPG_2011/comptage.shp"
)

```

## Graphique des résultats

Le graphique ggplot du nombre de parcelles comptées par ilot IRIS

```{r graphique}

comptage <- read_sf("RPG_2011/comptage.shp")

ggplot(comptage,aes(x=Nom_Iris,y=NUMPOINTS))+geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 30, hjust = 1, vjust = 1))+ggtitle("Nombre de parcelles RPG par ilot IRIS")


```
````

En installant le package R « Reticulate » on peut aussi insérer des chunk en python dans son RMD, on peut même partager les objets entre python et R

Après exportation par 

le fichier résultat est ci-dessous :

test_Rmarkdown

2024-10-17

Traitements QGIS pour le comptage de parcelles

Dans cette partie traitement géographique avec QGIS comptage des *parcelles RPG* par zone IRIS

```
centroids <- tempfile(fileext = ".shp")

results <- qgis_run_algorithm("native:centroids",
                              INPUT="RPG_2011/RPG_2011_MAJ.shp",
                              OUTPUT=centroids
)

results <- qgis_run_algorithm("native:countpointsinpolygon",
                              POINTS=centroids,
                              POLYGONS="Contours_Iris/carto/IRIS_extrait72.shp",
                              OUTPUT="RPG_2011/comptage.shp"
)
```

Graphique des résultats

Le graphique ggplot du nombre de parcelles comptées par ilot IRIS

```
comptage <- read_sf("RPG_2011/comptage.shp")

ggplot(comptage,aes(x=Nom_Iris,y=NUMPOINTS))+geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 30, hjust = 1, vjust = 1))+ggtitle("Nom
bre de parcelles RPG par ilot IRIS")
```